Multiobjective programming for type-2 hierarchical fuzzy inference trees

Article

Accepted Version

Multiobjective Programming for Type-2 Hierarchical Fuzzy Inference Trees

Varun Kumar Ojha, *Member, IEEE,* Václav Snášel, *Senior Member, IEEE,* and Ajith Abraham, *Senior Member, IEEE*

Abstract

This paper proposes a design of hierarchical fuzzy inference tree (HFIT). An HFIT produces an optimum tree-like structure. Specifically, a natural hierarchical structure that accommodates simplicity by combining several low-dimensional fuzzy inference systems (FISs). Such a natural hierarchical structure provides a high degree of approximation accuracy. The construction of HFIT takes place in two phases. Firstly, a nondominated sorting based multiobjective genetic programming (MOGP) is applied to obtain a simple tree structure (low model's complexity) with a high accuracy. Secondly, the differential evolution algorithm is applied to optimize the obtained tree's parameters. In the obtained tree, each node has a different input's combination, where the evolutionary process governs the input's combination. Hence, HFIT nodes are heterogeneous in nature, which leads to a high diversity among the rules generated by the HFIT. Additionally, the HFIT provides an automatic feature selection because it uses MOGP for the tree's structural optimization that accept inputs only relevant to the knowledge contained in data. The HFIT was studied in the context of both type-1 and type-2 FISs, and its performance was evaluated through six application problems. Moreover, the proposed multiobjective HFIT was compared both theoretically and empirically with recently proposed FISs methods from the literature, such as McIT2FIS, TSCIT2FNN, SIT2FNN, RIT2FNS-WB, eT2FIS, MRIT2NFS, IT2FNN-SVR, etc. From the obtained results, it was found that the HFIT provided less complex and highly accurate models compared to the models produced by most of the other methods. Hence, the proposed HFIT is an efficient and competitive alternative to the other FISs for function approximation and feature selection.

V K Ojha is with the Chair of Information Architecture, ETH Zurich, Zurich, Switzerland e-mail: ojha@arch.ethz.ch

V Snášel is with the Dept. of Computer Science, Technical University of Ostrava, Czech Republic, e-mail: vaclav.snasel@vsb.cz

A Abraham is with Machine Intelligence Research Labs, Washington, USA, e-mail: ajith.abraham@ieee.org

This work was supported by the IPROCOM Marie Curie initial training network, funded through the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA Grant Agreement No. 316555.

Manuscript received Month xx, yyyy; revised Month xx, yyyy.

Index Terms

Hierarchical fuzzy inference system, multiobjective genetic programming, differential evolution, approximation, feature selection

I. INTRODUCTION

A fuzzy inference system (FIS)—composed of a fuzzifier to fuzzify input information, an inference engine to infer information from a rule base (RB), and a defuzzifier to return crisp information—solves a wide range of problems that are ambiguous, uncertain, inaccurate, and noisy. An RB of an FIS is a set of rules of the form IF-THEN, i.e., the antecedent and the consequent form. The Takagi–Sugeno–Kang (TSK) is a widely used FIS model [1]. It embraces the IF-THEN form, where the antecedent part consists of type-1 fuzzy sets (T1FS) and/or type-2 fuzzy sets (T2FS), and the consequent part consists of real values or a linear/nonlinear function.

Type-1 FIS (T1FIS) and type-2 FIS (T2FIS) differ when it comes to the representation of the antecedent part and the consequent part of a rule, and T1FS and T2FS differ in the definitions of their membership functions. Unlike the crisp output of a T1FS membership function (MF) [2], the output of a T2FS MF is fuzzy in nature [3]. Such nature of the T2FS MFs is advantageous in processing uncertain information more effectively than with T1FS MFs [4]. Hence, a T2FIS can overcome the inability of a T1FIS to fully handle or accommodate the linguistic and numerical uncertainties associated with a changing and dynamic environment [5].

However, a T2FIS is computationally expensive because it has a larger number of parameters than a T1FIS, and it requires a type-reduction mechanism in its defuzzification part. The interval T2FIS (IT2FIS) reduces the computational cost by employing a simplified T2FS, known as interval T2FS (IT2FS) [4]. An IT2FS MF is bounded by a lower MF (LMF) and an upper MF (UMF), and the area between the LMF and UMF is called the *footprint of uncertainty* [4]. Then, a type-reducer reduces IT2FS to interval-valued T1FS. Subsequently, the output of IT2FIS is produced by averaging the intervals.

The construction and tuning of the rules are among the vital tasks in the optimization of an FIS, where the rule's construction is met by combining the fuzzy sets and the rule's tuning is met by adjusting the MF's parameters and the consequent part's parameters. Such a form of rule optimization is often achieved by mapping the rule's parameters onto a real-valued genetic vector, and it is known as the Michigan Approach [6]. Similarly, the construction/optimization of the RB is met by the genetic selection of the rules at the RB. Such a form of RB optimization

is often achieved by mapping the rules onto a binary-valued genetic vector [7], and it is known as the Pittsburgh Approach [8].

However, FIS optimization is not limited only to its mapping onto the genetic vector, but a structural/network-like implementation of FIS is often performed [9]. Additionally, TSK-based hierarchical self-organizing learning dynamics have also been proposed [10]. Moreover, several researchers have focused on the FIS and neural network (NN) integration and its parameter optimization using various learning methods including gradient-decent and the metaheuristic algorithms [11]–[14]. The summaries of such optimization paradigms are described as follows:

A self-constructing neural fuzzy inference network (SONFIN), proposed by Juang et al. [15], is a six layered network structure whose optimization begins with no rule and then rules are incrementally added during the learning process. SONFIN uses a clustering method to partition the input space that governs the number of rules extracted from the data, then the parameters (MF's arguments) of the determined SONFIN structure are tuned by the backpropagation algorithm. Later, in [16], SONFIN's concept was extended for the construction of T2FIS, where a self-evolving IT2FIS (SEIT2FNN) that implements a TSK-type FIS model was proposed, and the parameters of the evolved structure were tuned by using the Kalman-filtering algorithm. Additionally, a simplified type-reduction process for SEIT2FNN was proposed in [17]. Like SONFIN, in [18], a TSK-type FIS model, called a dynamic evolving neural-fuzzy inference system (DENFIS), was proposed, which evolved incrementally by choosing active rules from a set of rules and employed an evolving clustering method to partition the input space and the least-square estimator to optimize its parameters.

To overcome some limitations of the self-organizing fuzzy NN paradigm, Tung et al. [19] proposed a self-adaptive fuzzy inference network (SaFIN) that applied a categorical learning induced partitioning algorithm to eliminate two limitations: 1) the need for predefined numbers of fuzzy clusters and 2) the stability–plasticity trade-off that addresses the difficulty in finding a balance between past knowledge and current knowledge during the learning process. SaFIN also employed a rule consistency checking mechanism to avoid inconsistent RB construction. Additionally, the Levenberg-Marquardt method was applied for RB's parameters tuning. In [20], to improve the efficiency of IT2FIS, a mutually recurrent interval type-2 neural fuzzy system (MRIT2NFS) was proposed which used weighted feedback loops in the antecedent parts of the formed rules and applied gradient-decent learning and a Kalman-filter algorithm to tune the recurrent weights and the rules' parameters, respectively. In [21], a self-evolving T2FIS model

was proposed that employed a compensatory operator in the type-2 inference mechanism and a variable-expansive Kalman-filter algorithm for parameter tuning.

Further, a simplified interval type-2 fuzzy NN with a simplified type-reduction process (SIT2FIS) was proposed in [22], and a growing online self-learning IT2FIS that used the dynamics of a growing Gaussian mixture model was proposed in [23]. Recently, in [24], a meta-cognitive interval type-2 neuro FIS (McIT2FIS) was proposed, which employs a self-regulatory meta-cognitive system that extracts the knowledge contained in minimal samples by accepting or discarding data samples based on sample's contribution to knowledge. For the parameters tuning, McIT2FIS employed the Kalman-filtering algorithm.

However, the self-organizing fuzzy NN paradigm discussed above has to employ a clustering method to partition the input space during the FIS structure's design. Contrary to this, a hierarchical FIS (HFIS) constructs an FIS by using a hierarchical arrangement of several low-dimensional fuzzy subsystems [25]. Initially, the input variables selection, the levels of hierarchy, and the number of parameters was fully up to the experts to determine. Moreover, HFIS design overcomes the *curse of dimensionality* [26], and it possesses a universal approximation ability [27]–[30].

Torra et al. [31] summarized the contributions where the expert's role in the HFIS design process was minimized/eliminated. For example, in [32], HFIS was realized as a feedforward network like structure in which the output of the previous layer's subsystem was only fed to the consequent part of the next layer, and so on. Similarly, in [33], a two-layered HFIS was developed, where, for each layer, the knowledge bases (KB) were generated by linguistics rule generation method and the KB rules were selected by genetic algorithm (GA). In [34], an adaptive fuzzy hierarchical sliding-mode control method was proposed, which was an arrangement of many subsystems, and the top layer accommodated all the subsystems' outputs. Moreover, in [35], to optimize the structure of a hierarchical arrangement of low-dimensional TSK-type FISs, probabilistic incremental program evolution [36] was employed. Similarly, the importance of the hierarchical arrangements of the low-dimensional T2FSs is explained in [5], [37].

For FIS models that have a structural representation (e.g., self-organizing fuzzy NN and HFIS models), multiobjective optimization is inherent since accuracy maximization and complexity minimization are two desirable objectives [38]. Hence, to make trade-offs between interpretability and accuracy, or, in other words, to make trade-offs between approximation error minimization and complexity minimization, a multiobjective orientation of FIS optimization can be used [39]–[41]. Complexity minimization can be defined in many ways, such as a reduced number of rules,

reduced number of parameters, etc. [41], [42].

Since a single solution may not satisfy both objectives simultaneously, a Pareto-based multiobjective optimization algorithm can be used in FIS optimization, the scope of which spans from the rule selection, to rule mining, rule learning, etc. [43]–[46]. Similarly, in [47]–[50], simultaneous learning of KB was proposed, which included feature selection, rule complexity minimization together with approximation error minimization, etc.

Moreover, in [51], a co-evolutionary approach that aimed at combining a multiobjective approach with a single objective approach was presented where, at first, a multiobjective GA determined a Pareto-optimal solution by finding a trade-off between accuracy and rule complexity. Then, a single objective GA was applied to reduce training instances. Such a process was then repeated until a satisfactory solution was obtained. A summary of research works focused on multiobjective optimization of FIS is provided in [52].

In conclusion, the following are the necessary practices for an FIS model design: 1) input space partitioning; 2) rule formation; 3) rule tuning; 4) FIS structural representation; 5) improving accuracy and minimizing a model's complexity. Therefore, in this work, a multiobjective optimization of HFIS, called a hierarchical fuzzy inference tree (HFIT), was proposed.

Unlike the self-organizing paradigm that has a network-like structure and uses a clustering algorithm for partitioning of input space, the proposed HFIT constructs a tree-like structure and uses the dynamics of the evolutionary algorithm for partitioning input space [53]. The HFIT is analogous to a multi-layered network and automatically partitions input space during the structure optimization phase, i.e., during the tree construction phase. The parameter tuning of the HFIT was performed by the differential evolution (DE) algorithm [54], which is a metaheuristic algorithm inspired by the dynamics of the evolutionary process. Metaheuristic algorithms, being independent of the problems, solve complex optimization problems. Hence, they are useful in finding the appropriate parameter values for an FIS [13].

In this work, the proposed HFIT implements a TSK-type FIS for both T1FIS and T2FIS, and HFIT was studied under both single objective and multiobjective optimization orientations. Hence, a total of four versions of HFIT algorithms were proposed: type-1 single objective HFIT (T1HFIT$^{\mathrm{S}}$), type-1 multiobjective objective HFIT (T1HFIT$^{\mathrm{M}}$), type-2 single objective HFIT (T2HFIT$^{\mathrm{S}}$), and type-2 multiobjective objective HFIT (T2HFIT$^{\mathrm{M}}$). In the construction of type-2 HFITs, the type-reduction algorithm of the Karnik-Mendel method described in [4] was used with an improvement in its termination criteria. In summary, the following are the main and

novel contributions of this work.

1) The proposed hierarchical tree-like design (HFIT) forms a natural hierarchical structure by combining several low-dimensional fuzzy subsystems.
2) MOGP driven optimization provided a trade-off between model's accuracy and complexity. Moreover, in the obtained tree, each node has a different input's combination, where the MOGP governs the input's combination. Hence, HFIT nodes are heterogeneous in nature, which leads to a high diversity among the rules generated by the HFIT. Such a diverse rule generation methods is a distinguished aspect of the proposed HFIT.
3) A comprehensive theoretical study of HFIT shows that when it comes to the partitioning of input space, membership function design, and even rule formation, it has advantages over network-like layered architecture models, which have to use clustering methods when they do input space partitioning. Clustering methods generate overlapping MFs in fuzzy sets, whereas HFIT's MOGP driven MFs selection avoid such a overlapping of MFs.
4) Unlike many models in the literature, HFIT performed an inclusive automatic feature selection, which led to the simplification of the RB in fuzzy subsystems and incorporated only relevant knowledge contained in the dataset into HFIT's structural representation.
5) A comprehensive performance comparison of the proposed four versions of the HFIT algorithms both in theoretical and empirical sense with the recently proposed FIS algorithms found in the literature suggests that HFIT design offers a high approximation ability with simple model complexity.

The structure of this article is as follows: Section II provides an introduction to T1FIS and T2FIS; Section III describes the proposed multiobjective strategy for developing HFIT and its parameter optimization; Section IV provides a comprehensive theoretical evaluation of HFIT; Section V provides a detailed description of parameter setting and a comprehensive empirical evaluation the proposed HFIT compared with the algorithms reported in the literature; finally, the obtained results are discussed in Section VI followed by a concise conclusion in Section VII.

II. TSK Fuzzy Inference Systems

A. Type-1 Fuzzy Inference Systems

A TSK-type FIS is governed by the IF–THEN rules of the form [1]:

$$R_i : \text{IF } x_1 \text{ is } A_{i1} \text{ AND } \ldots \text{ AND } x_{d^i} \text{ is } A_{id^i} \text{ THEN } y \text{ is } B_i \tag{1}$$

where R_i is the i-th rule in an FIS, $A_{i1}, \ldots, A_{id^i}$ are the T1FSs, B_i is a function of an input vector $\mathbf{x} = \langle x_1, x_2, \ldots, x_{d^i} \rangle$ that returns a crisp output y, and d^i is the total number of the inputs presented to the i-th rule. Note that the number of inputs may vary from rule-to-rule. Hence, the dimension of inputs in a rule is denoted as d^i. In TSK, the function B_i is usually expressed as:

$$B_i = c_i^0 + \sum_{j=1}^{d^i} c_i^j x_j \tag{2}$$

where c_i^j for j = 0 to d^i is the free parameters in the consequent part of a rule. The defuzzified crisp output of FIS is computed as follows: First, the inference engine fires up the RB rules. The firing strength f_i of the i-th rule is computed as:

$$f_i = \prod_{j=1}^{d^i} \mu_{A_{ij}}(x_j) \tag{3}$$

where $\mu_{A_{ij}}$ is the value of j-th T1FS MF at the i-th rule. Then, the defuzzified output y of an FIS is computed as:

$$y = \frac{\sum_{i=1}^{M} B_i f_i}{\sum_{i=1}^{M} f_i} \tag{4}$$

where M is the total rules in the RB. In this work, as shown in Fig. 1(a), the T1FS A was of the form:

$$\mu_A(x) = \frac{1}{1 + \left(\frac{x-m}{\sigma}\right)^2} \tag{5}$$

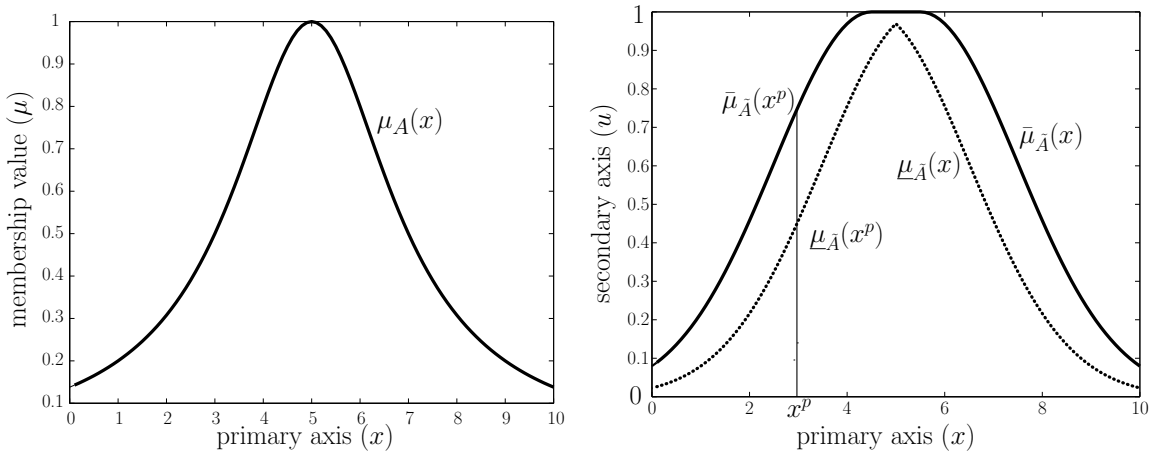
where m and σ are the center and the width of MF $\mu_A(x)$, respectively.

B. Type-2 Fuzzy Inference Systems

A T2FS $\tilde{A}$ is characterized by a 3-dimensional (3-D) MF [55]. The three axes of T2FS are defined as follows. The x-axis is called the primary variable, the y-axis is called the secondary variable (or primary MF, which is denoted by u), and the z-axis is called the MF value (or secondary MF value), which is denoted by μ. Hence, in a universal set X, a T2FS $\tilde{A}$ has the form:

$$\tilde{A} = \{((x,u), \mu_{\tilde{A}}(x,u)) \,|\forall x \in X, \forall u \in [0,1]\} \tag{6}$$

where the MF value μ has a 2-dimensional support called the *footprint of uncertainty* of $\tilde{A}$, which is bounded by an LMF $\underline{\mu}_{\tilde{A}}(x)$ and a UMF $\bar{\mu}_{\tilde{A}}(x)$ (Fig. 1(b)). A Gaussian function, with



(a) Type-1 Fuzzy MF (b) Type-2 Fuzzy MF

Fig. 1. Membership functions. (a) Type-1 MF (5) with mean $m = 5.0$ and $\sigma = 2.0$ (b) Type-2 Fuzzy MF with fixed $\sigma = 2.0$ and means $m_1 = 4.5$ and $m_2 = 5.5$. LMF $\underline{\mu}_{\tilde{A}}(x)$ as per (8) is on the dotted line and UMF $\bar{\mu}_{\tilde{A}}(x)$ as per (9) is on the solid line.

an uncertain mean within $[m_1, m_2]$ and standard deviation σ, is an IT2FS MF (Fig. 1(b)), which is written as:

$$\mu_{\tilde{A}}(x, m, \sigma) = \exp\left(-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2\right), \qquad m \in [m_1, m_2]. \tag{7}$$

In this work, the LMF was defined as [4]:

$$\underline{\mu}_{\tilde{A}}(x) = \begin{cases} \mu_{\tilde{A}}(x, m_2, \sigma), & x \leq (m_1 + m_2)/2 \\ \mu_{\tilde{A}}(x, m_1, \sigma), & x > (m_1 + m_2)/2 \end{cases} \tag{8}$$

and the UMF was defined as [4]:

$$\bar{\mu}_{\tilde{A}}(x) = \begin{cases} \mu_{\tilde{A}}(x, m_1, \sigma), & x < m_1 \\ 1, & m_1 \leq x \leq m_2 \\ \mu_{\tilde{A}}(x, m_2, \sigma), & x > m_2 \end{cases}. \tag{9}$$

In Fig. 1(b), the point x^p along the x-axis of 3-D IT2FS MF cuts the LMF and UMF along the y-axis, and the value of the IT2FS is considered to be along the z-axis (not shown in Fig. 1(b)) are $\bar{\mu}_{\tilde{A}}(x^p)$ and $\underline{\mu}_{\tilde{A}}(x^p)$. Considering IT2FS MFs, i-th IF–THEN rule of type-2 TSK-FIS for an input vector $\mathbf{x} = \langle x_1, x_2, \ldots, x_{d^i} \rangle$ takes the following form:

$$R_i : \text{IF } x_1 \text{ is } \tilde{A}_{i1} \text{ AND } \ldots \text{ AND } x_{d^i} \text{ is } \tilde{A}_{id^i} \text{ THEN } y \text{ is } \tilde{B}_i \tag{10}$$

where $\tilde{A}_{i1}, \ldots, \tilde{A}_{id^i}$ are the T2FSs, $\tilde{B}_i$ is a function of $\mathbf{x}$ that returns a pair $[\underline{b}_i, \bar{b}_i]$ called the left and right weights of the consequent part of the i-th rule. In TSK, $\tilde{B}_i$ is usually written as:

$$\tilde{B}_i = [c_i^0 - s_i^0, c_i^0 + s_i^0] + \sum_{j=1}^{d^i} [c_i^j - s_i^j, c_i^j + s_i^j] x_j \tag{11}$$

where c_i^j for j = 0 to d^i is the free parameter in the consequent part of a rule and s_j^i for j = 0 to d^i are the deviation factors of the free parameters. The firing strength of IT2FS $F_i = [\underline{f}_i, \bar{f}_i]$ is computed as:

$$\underline{f}_i = \prod_{j=1}^{d^i} \underline{\mu}_{\tilde{A}_{ij}}(x_j) \quad \text{and} \quad \bar{f}_i = \prod_{j=1}^{d^i} \bar{\mu}_{\tilde{A}_{ij}}(x_j) \tag{12}$$

At this stage, inference engine fires up the rule and the type-reducer reduces the IT2FS to T1FS. In this work, the *center of set* type-reducer y_{cos}, prescribed in [4], was used:

$$y_{cos} = \bigcup_{f^i \in F^i,\ b^i \in \tilde{B}^i} \frac{\sum_{i=1}^{M} f^i b^i}{\sum_{i=1}^{M} f^i} = [y_l, y_r] \tag{13}$$

where y_l and y_r are the left and the right end of the interval. For the ascending order of $\underline{b}^i$ and $\bar{b}^i$, y_l and y_r are computed as:

$$y_l = \frac{\sum_{i=1}^{L} \bar{f}^i \underline{b}^i + \sum_{i=L+1}^{M} \underline{f}^i \underline{b}^i}{\sum_{i=1}^{L} \bar{f}^i + \sum_{i=L+1}^{M} \underline{f}^i} \tag{14}$$

$$y_r = \frac{\sum_{i=1}^{R} \underline{f}^i \bar{b}^i + \sum_{i=R+1}^{M} \bar{f}^i \bar{b}^i}{\sum_{i=1}^{R} \underline{f}^i + \sum_{i=R+1}^{M} \bar{f}^i} \tag{15}$$

where L and R are the switch point, determined by

$$\underline{b}^L \leq y_l \leq \underline{b}^{L+1} \quad \text{and} \quad \bar{b}^R \leq y_r \leq \bar{b}^{R+1},$$

respectively. The defuzzified crisp output is then computed as:

$$y = \frac{y_l + y_r}{2}. \tag{16}$$

III. Multiobjective Optimization of Hierarchical Fuzzy Inference Trees

A. Hierarchical Tree Formation

A hierarchical fuzzy inference tree (HFIT) is a tree-based system. Its hierarchical structure is analogous to a multilayer feedforward NN, where the nodes (the low-dimensional FISs) are connected using weighted links. The concept of forming a hierarchical fuzzy inference tree is inherited from the flexible neural tree proposed by Chen et al. [56], which has two

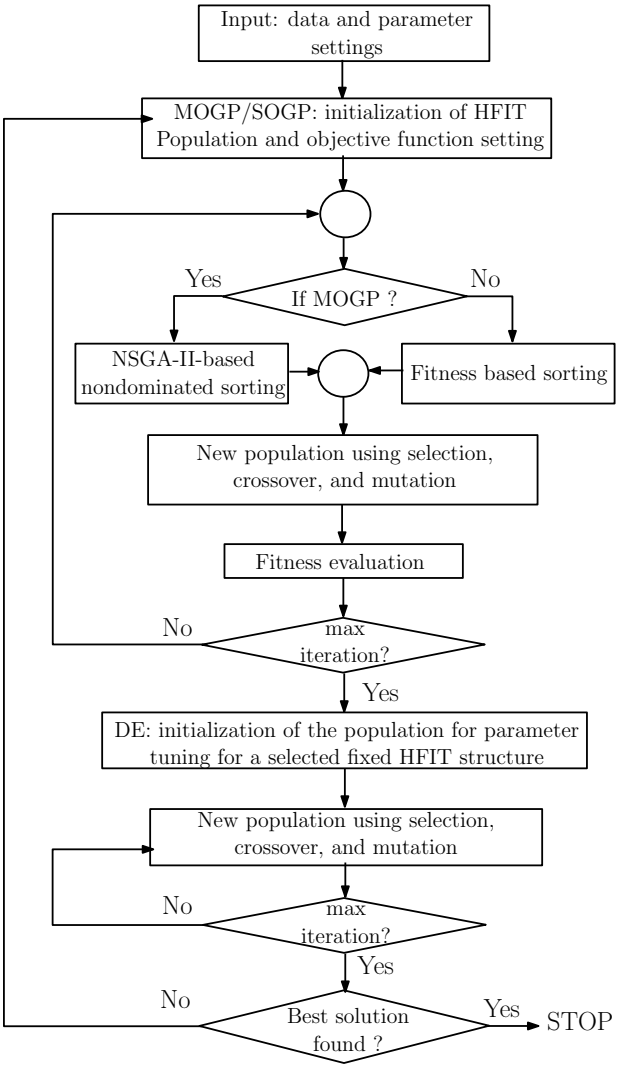
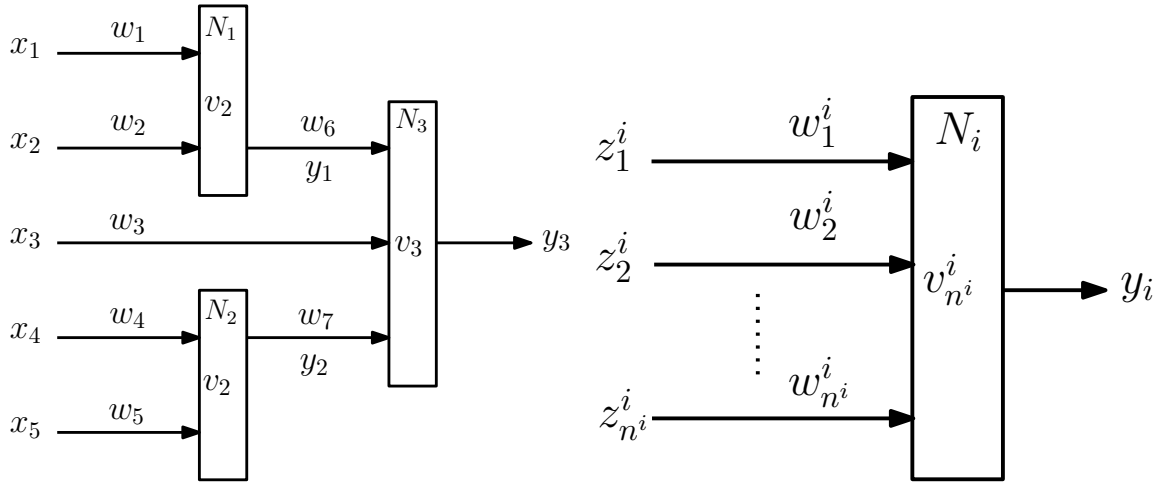


Fig. 2. Two-phase construction of a hierarchical fuzzy inference tree.

learning phases. First, in the *tree construction* phase, an evolutionary algorithm is employed to construct/optimize a tree-like structure. Second, in the *parameter tuning* phase, a genotype representing the underlying parameters of the tree structure is optimized by using parameter optimization algorithms.

To create an optimum tree based model, firstly, a population of randomly generated trees is formed. Once a satisfactory tree structure (a tree with a small approximation error and low complexity) is obtained using an evolutionary algorithm, the *parameter tuning* phase optimizes its parameters. The phases are repeated until a satisfactory solution is obtained. Fig. 2 is a clear representation of HFIT's two-phase construction approach.



(a) Two-stage hierarchical tree. (b) Node structure.

Fig. 3. Hierarchical fuzzy inference system. (a) Complete tree with three nodes N_1, N_2, and N_3 and with inputs x_1, x_2, x_3, x_4, and x_5. (b) Illustration of the i-th node N_i that has n^i inputs $z_j^i \in [x_1, \ldots, x_n]$ for $j = 1$ to n^i and output y_i.

B. Tree Encoding

An HFIT G is a collection of nodes V and terminal nodes T:

$$G = V \cup T = \{v_2, v_3, \ldots, v_{tn}\} \cup \{x_1, x_2, \ldots, x_d\} \tag{17}$$

where v_j $(j = 2, 3, \ldots, tn)$ denotes non-leaf instruction and has $2 \leq j \leq tn$ arguments. The leaf node's instruction $x_1, x_2, \ldots, x_d$ takes no argument and represents the input variable/instruction. A typical HFIT is shown in Fig. 3(a); whereas, Fig. 3(b) illustrates an HFIT's node N_i that takes n^i inputs. The inputs $z_j^i \in \{x_1, x_2, \ldots, x_d\}$ for $j = 1$ to n^i to a node N_i is either from the input layer or from other nodes in HFIT. Each node in an HFIT receives a weighted input $x_i w_i$, where w_i is the weight. In this work, however, the weights in HFIT were set to 1 because the objective of this work was also to reduce the complexity of the produced tree along with approximation error. Setting weights to 1 also allow raw input to be fed to the fuzzy sets.

C. Rule Formation at the Nodes

1) Rules for Type-1 FIS Node: Each node in an HFIT is an FIS of either type-1 or type-2. Hence, the rules at a node were created as follows: Considering a reference to the node N_1 from

Fig. 3(a) that has two arguments/inputs x_1 and x_2 and assuming that each input x_1 and x_2 has two T1FSs A^1_{11}, A^1_{12} and A^1_{21}, A^1_{22}, respectively, the rules for T1FIS are generated as:

$$R^1_{ij}: \text{IF } x_1 \text{ is } A^1_{1i} \text{ AND } x_2 \text{ is } A^1_{2j} \text{ THEN } y^1_{ij} = c^0_{ij} + c^1_{ij}x_1 + c^2_{ij}x_2,$$
$$\text{for } i = 1, 2 \text{ and } j = 1, 2.$$

The consequent part B^1_{ij} of the rules at the node N_1 is computed by using (2). Finally, the output y_1 of node N_1 is computed as:

$$y_1 = \frac{\sum_{i=1}^{2}\sum_{j=1}^{2} f^1_{ij}B^1_{ij}}{\sum_{i=1}^{2}\sum_{j=1}^{2} f^1_{ij}} \tag{18}$$

where the firing strength f^1_{ij} is computed as:

$$f^1_{ij} = \mu_{A^1_{1i}}(x_1)\mu_{A^1_{2j}}(x_2), \qquad \text{for } i = 1, 2 \text{ and } j = 1, 2. \tag{19}$$

Similar to node N_1, node N_2 has two inputs and, if each input at node N_2 is partitioned into two T1FSs, then the output y_2 of node N_2 is computed in a similar way to how the output of the node N_1 is computed.

The output y_3 of the HFIT shown in Fig. 3(a) is computed from node N_3, which revives inputs y_1 and y_2 and x_3, where y_1 and y_2 are the outputs of nodes N_1 and N_2, respectively. Therefore, the rules at node N_3, considering each input y_1, y_2, and x_3 has two T1FSs A^3_{11}, A^3_{12}, A^3_{21}, A^3_{22}, and A^3_{31}, A^3_{32} respectively, is represented as:

$$R^3_{ijk}: \text{IF } y_1 \text{ is } A^3_{1i} \text{ AND } y_2 \text{ is } A^3_{2j} \text{ AND } x_3 \text{ is } A^3_{3k} \text{ THEN } y^3_{ijk} = c^0_{ijk} + c^1_{ijk}y_1 + c^2_{ijk}y_2 + c^3_{ijk}x_3,$$
$$\text{for } i = 1, 2 \text{ and } j = 1, 2 \text{ and } k = 1, 2.$$

Output y_3 of node N_3, which is also the output of the tree (Fig. 3(a)), is computed as:

$$y_3 = \frac{\sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{k=1}^{2} f^3_{ijk}B^3_{ijk}}{\sum_{i=1}^{2}\sum_{j=1}^{2}\sum_{j=1}^{2} f^3_{ijk}} \tag{20}$$

where the consequent part B^3_{ij} is computed using (2) and the firing strength f^3_{ijk} is computed as:

$$f^3_{ij} = \mu_{A^3_{1i}}(y_1)\mu_{A^3_{2j}}(y_2)\mu_{A^3_{3k}}(x_3), \qquad \text{for } i = 1, 2 \text{ and } j = 1, 2 \text{ and } k = 1, 2. \tag{21}$$

2) Rules for Type-2 FIS Node: If the nodes of the HFIT in Fig. 3(a) are type-2 nodes, then, assuming that node N_1 has two T2FSs $\tilde{A}^1_{11}, \tilde{A}^1_{12}$ and $\tilde{A}^1_{21}, \tilde{A}^1_{22}$, respectively, the rules for T2FIS at node N_1 are generated as:

R^1_{ij} : IF x_1 is $\tilde{A}^1_{1i}$ AND x_2 is $\tilde{A}^1_{2j}$ THEN $y^1_{ij} = [c^0_{ij} - s^0_{ij}] + [c^1_{ij} - s^1_{ij}]x_1 + [c^2_{ij} - s^2_{ij}]x_2$,
for $i = 1, 2$ and $j = 1, 2$

and the lower and upper firing strengths $\underline{f}^1_{ij}$ and $\bar{f}^1_{ij}$ at node N_1 are computed as:

$$\underline{f}^1_{ij} = \mu_{\tilde{A}^1_{1i}}(x_1)\mu_{\tilde{A}^1_{2j}}(x_2), \qquad \text{for } i = 1, 2 \text{ and } j = 1, 2 \tag{22}$$

$$\bar{f}^1_{ij} = \mu_{\tilde{A}^1_{1i}}(x_1)\mu_{\tilde{A}^1_{2j}}(x_2), \qquad \text{for } i = 1, 2 \text{ and } j = 1, 2. \tag{23}$$

Then, the left and right weights $\underline{b}^1_{ij}$ and $\bar{b}^1_{ij}$ of the consequent part of the rules are produced by using (11). Thereafter, the type-reduction of the node is performed as described in [4], where the left and right intervals y^1_l and y^1_r are computed as per (14) and (15). During type-reduction [4], an early stopping mechanism was adopted to reduce computation time. Finally, output y_1 of node N_1 is computed as $y_1 = (y^1_l + y^1_r)/2$.

The output computation at node N_2 of the tree in Fig. 3(a) is similar to that of the output computation of node N_1 because, at node N_2, there are two inputs and each of these are partitioned into two T2FSs.

The output of the type-2 HFIT shown in Fig. 3(a) is computed from node N_3, which receives inputs y_1 and y_2 and x_3, where y_1 and y_2 are the outputs of nodes N_1 and N_2, respectively. Therefore, the rules at node N_3, considering each input y_1, y_2, and x_3 has two T2FSs $\tilde{A}^3_{11}, \tilde{A}^3_{12}$, $\tilde{A}^3_{21}, \tilde{A}^3_{22}$, and $\tilde{A}^3_{31}, \tilde{A}^3_{32}$ respectively, are represented as:

R^3_{ijk} : IF y_1 is $\tilde{A}^3_{1i}$ AND y_2 is $\tilde{A}^3_{2j}$ AND x_3 is $\tilde{A}^3_{3k}$ THEN
$y^3_{ijk} = [c^0_{ijk} - s^0_{ijk}] + [c^1_{ijk} - s^1_{ijk}]y_1 + [c^2_{ijk} - s^2_{ijk}]y_2 + [c^3_{ijk} - s^3_{ijk}]x_3$,
for $i = 1, 2$ and $j = 1, 2$ and $k = 1, 2$.

The lower and upper firing strengths $\underline{f}^3_{ijk}$ and $\bar{f}^3_{ijk}$ at node N_3 are computed as:

$$\underline{f}^3_{ijk} = \mu_{\tilde{A}^3_{1i}}(y_1)\mu_{\tilde{A}^3_{2j}}(y_2)\mu_{\tilde{A}^3_{3j}}(x_3), \text{ for } i = 1, 2 \text{ and } j = 1, 2 \text{ and } k = 1, 2 \tag{24}$$

$$\bar{f}^3_{ijk} = \mu_{\tilde{A}^3_{1i}}(y_1)\mu_{\tilde{A}^3_{2j}}(y_2)\mu_{\tilde{A}^3_{3j}}(x_3), \text{ for } i = 1, 2 \text{ and } j = 1, 2 \text{ and } k = 1, 2. \tag{25}$$

After computing the firing strengths and the left and right weights $\underline{b}^3_{ij}$ and $\bar{b}^3_{ij}$ of the rules, the type-reduction at the node is performed by using (13), where the left and right intervals y^3_l and y^3_r are computed as per (14) and (15). Output y_3 of node N_3, which is also the output of the tree, is computed by averaging y^3_l and y^3_r as $y_3 = (y^3_l + y^3_r)/2$.

D. Structure Tuning (Pareto-based Multiobjective Optimization)

Usually, a learning algorithm owns a single objective (approximation error minimization) that is often achieved by minimizing the root mean squared error (RMSE) on the learning data:

$$E = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i - y_i)^2} \tag{26}$$

where d and y are the desired and the model's outputs, respectively, and N is the number of data pairs in the training set. However, a single objective comes at the expense of a model's complexity or the generalization ability on unseen data. The generalization ability broadly depends on the model's complexity (e.g., the number of parameters $c(\mathbf{w})$ in the model) [57]. The minimization of the approximation error E and the number of free parameters $c(\mathbf{w})$ are conflicting objectives. Hence, a Pareto-based multiobjective optimization can be applied to obtained a Pareto set of nondominated solutions, in which no one objective function can be improved without a simultaneous detriment to at least one of the other objectives of the solution [58]

Therefore, an HFIT that offers the lowest approximation error and simplest structure is the most desirable one. To obtain such a set of Pareto-optimal (nondominated) solutions, a nondominated sorting based MOGP was applied.

The proposed MOGP acquires the nondominated sorting algorithm [58] for computing Pareto-optimal solutions from an initial population of fuzzy inference trees. The individuals in MOGP were sorted according to their dominance in population. Moreover, individuals were sorted according to the rank/Pareto-front/line. MOGP is an elitist algorithm that allows the best individuals to propagate into the next generation. Diversity in population was maintained by measuring the crowding distance among the individuals [58].

A detailed description of MOGP algorithm is as follows:

1) Initial Population: Two fitness measures were considered: *approximation error* minimization and *parameter count* minimization. To simultaneously optimize these objectives during the *structure-tuning* phase, an initial population W^0 of randomly generated HFITs was formed and sorted according to their nondominance.

2) Selection: In selection operation, a *mating pool* W^p of $size(\mathrm{W}^0)/2$ was obtained using *binary tournament selection* that selects two candidates randomly at a time from a population W^t, and the best solution (according to its rank and crowding distance) is copied into the mating pool W^p. This process is continued until the mating pool becomes full.

3) Generation: An offspring population W^c was generated using the individuals of the mating pool W^p. Two distinct individuals (parents) were randomly selected from the mating pool to create new individuals using the genetic operators crossover and mutation.

4) Crossover: In crossover operation, randomly selected sub-trees of two parent trees are swapped (Fig. 4(a)). The swapping includes the exchange of nodes. A detailed description of the crossover operation in genetic programming is available in [59], [60]. The crossover operation is selected with the crossover probability pc.

5) Mutation: The mutation operators used in HFIT are as follows [59], [60]:

a) Replacing a randomly selected terminal $x_i \in T$ with a newly generated terminal $x_j \in T$ for $j \neq i$.
b) Replacing all terminal nodes of an HFIT with a new set of terminal nodes derived from T.
c) Replacing a randomly selected FIS node $N_i \in F$ with a newly generated FIS node $N_j \in F$ for $j \neq i$.
d) Replacing a randomly selected terminal node $x_i \in T$ with a newly created FIS node $N_i \in F$.
e) Deleting a randomly selected terminal node $x_i \in T$ or deleting a randomly selected FIS node $N_i \in F$.

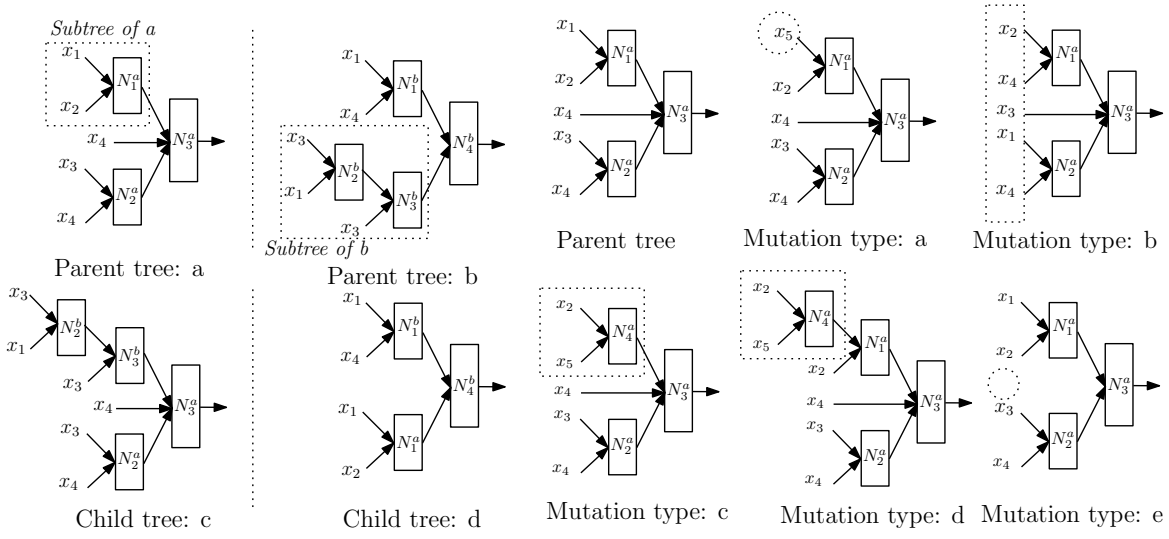
The mutation operation was selected with the probability pm, and the type of mutation operator (a or b or c or d or e) was chosen randomly during the mutation operation (Fig. 4(b)).

6) Recombination: The offspring population W^c and the main population W^t were mixed together to make a combined population W^g.

7) Elitism: In this work, elitism was decided according to the rank (based on both RMSE and the model's complexity) of the individuals (HFITs) in the population. Therefore, in this step, $size(\mathrm{W}^c)$ worst (poorer rank) individuals were weeded out from the combined population W^g. In other words, $size(\mathrm{W}^t)$ best individuals are propagated into the new generation $t+1$ as the main population W^{t+1}.

E. Parameter Tuning

In the structure tuning phase, an optimum phenotype (HFIT) was derived with the parameters being initially fixed by random guesswork. Hence, the obtained phenotype was further tuned in the parameter tuning phase by using a parameter optimization algorithm. To tune the parameters of the derived phenotype, its parameters were mapped onto a genotype, i.e., onto a real vector, called a solution vector. The selection of the best phenotype in a single objective training was solely based on a comparison of the RMSEs. However, selecting a solution in a multiobjective training is a difficult choice. In this work, after the multiobjective training of HFIT, the best solution for parameter tuning was picked from the Pareto front. Strictly, the solution that gave



(a) Crossover operation (b) Mutation operation

Fig. 4. MOGP Operations: (a) Crossover between two parent trees a and b. (b) A total five types of mutation of the parent tree.

the best RMSE among the solutions marked rank-one in the Pareto-optimal set was chosen. Fig. 5 is an illustration of the solutions that belong to the Pareto-front.

The genotype mapping of a T1FIS and a T2FIS differ only in regard to their number of parameters. In HFIT, a T1FIS uses the MF mentioned in (5), which has two arguments m and σ and each rule in T1FIS has $d^i + 1$ variables in the consequent part as referred to in (2), where d^i is the number of inputs to the i-th rule. On the other hand, a T2FIS uses IT2FSs, which are bounded by LMFs and UMFs (Fig. 1(b)) and have two Gaussian means m_1 and m_2 and a variance σ to be optimized. The Gaussian means m_1 and m_2 for type-2 Gaussian MF (7) were defined as:

$$m_1 = m + \lambda\sigma \quad \text{and} \quad m_2 = m - \lambda\sigma,$$

where $\lambda \in [0, 1]$ is a random variable taken from uniform distribution and m is the center of the Gaussian means m_1 and m_2 taken from [0, 1]. Similarly, the variance σ of type-2 Gaussian MF (7) was taken from [0, 1]. The consequent part of the T2FIS was computed according to (11), which led to $2(d^i + 1)$ variables.

Assume that an HFIT (a tree like Fig. 3(a)) has k many nodes and each node in the phenotype takes $2 \leq d^i \leq tn$ inputs, where each input is partitioned into two fuzzy sets (MFs). Then, the number of the fuzzy sets at a node is $2d^i$. Since the number of inputs at a node is d^i and each input is partitioned into two fuzzy sets, the number of rules at a node is 2^{d^i}. Hence, the number

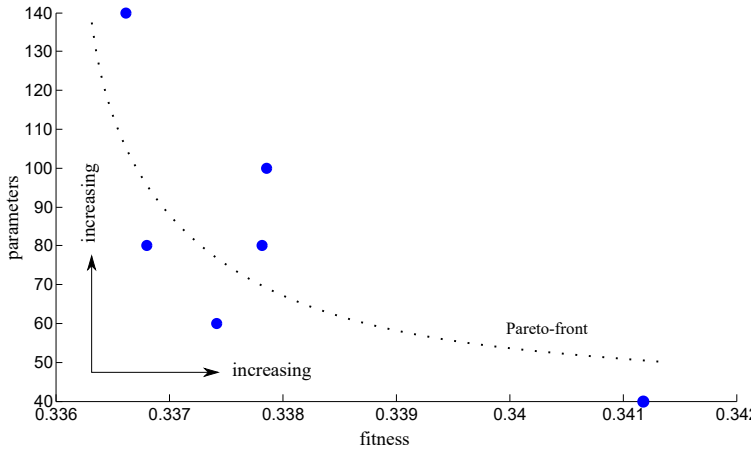


Fig. 5. FIS fitness versus FIS parameter mapping across the Pareto-front. This graph was created during a multiobjective training of the example–2 mentioned in Section V-B.

of parameters at a T1FIS node is $[2(2d^i) + 2^{d^i}(d^i + 1)]$ and the number of parameters at a T2FIS node is $[3(2d^i) + 2^{d^i}(2(d^i + 1))]$. Therefore, the total number of parameters in an HFIT is the summation of the number of parameters at all k nodes in the tree. For example, the number of parameters in the type-1 HFIT and type-2 HFIT shown in Fig. 3(a) are 84 and 154 respectively.

Assuming n is the total number of parameters in a tree, the genotype or the solution vector $\mathbf{w}$ corresponding to the tree (phenotype) is expressed as:

$$\mathbf{w} = \langle w_1, w_2, \ldots, w_n \rangle \tag{27}$$

Now, to optimize parameter vector $\mathbf{w}$, a parameter optimizer can be used: genetic algorithms [59], evolution strategy [59], artificial bee colony [61], PSO [62], DE [63], gradient-based algorithms [64], backpropagation [65], Klaman-filter [66], etc.

In this work, the differential evolution (DE) version "DE/rand-to-best/1/bin" [54] was used, which is a metaheuristic algorithm that uses a crossover operator inspired by the dynamics of "natural selection." The basic principle of the DE is as follows: First, an initial population matrix $\mathrm{W}^t = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_P)$ at the iteration $t = 0$ is randomly initialized. The population W^t contains P many solution vectors. A solution vector $\mathbf{w}$ in the population is an n-dimensional vector representing the free parameters of an HFIT. Secondly, the population W^{t+1} is created using binomial trials. Hence, to create a new solution vector for the population W^{t+1}, three distinct solution vectors $\mathbf{w}^a$, $\mathbf{w}^b$, and $\mathbf{w}^c$ and the best solution vector $\mathbf{w}^g$ are selected from the population

W^t. Then, for a random index $k \in [1, n]$ and for the selected trial vector $\mathbf{w}^a = \langle w_1^a, w_2^a, \ldots, w_n^a \rangle$, the j-th variable of the modified trial vector $\mathbf{w}^{a'}$ are created as:

$$w_j^{a'} = \begin{cases} w_j^a + F(w_j^g - w_j^a) + F(w_j^b - w_j^c), & r_j < cr \parallel j = k \\ w_j^a, & r_j \geq cr \end{cases} \tag{28}$$

where $r_j \in [0, 1]$ is a uniform random sample, $cr \in [0, 1]$ is the crossover rate, and $F \in [0, 2]$ is the differential weight. Similarly, all the variables $j = 1$ to n of the trial vector $\mathbf{w}^a$ is created using (28). After creation of the modified trial vector $\mathbf{w}^{a'}$, it is *recombined* as:

$$\mathbf{w}^a = \begin{cases} \mathbf{w}^{a'}, & E(\mathbf{w}^{a'}) < E(\mathbf{w}^a) \\ \mathbf{w}^a, & E(\mathbf{w}^{a'}) \geq E(\mathbf{w}^a) \end{cases} \tag{29}$$

where $E(.)$ is the function that returns the fitness of a solution vector using (26). In DE , operators, such as *selection*, *crossover*, and *recombination* were repeated until a satisfactory solution vector $\mathbf{w}^*$ was found or no improvement was observed compared to an obtained solution over a fixed period (100 DE iterations).

IV. Theoretical Evaluation

Efficiency of the proposed HFIT comes from a combined influence of three basic operations involved in the model's development: tree construction through MOGP, combining several low-dimensional fuzzy systems in a hierarchical manner, and parameters tuning through differential evolution (DE). Hence, HFIT bears many distinguished properties that define its prediction efficiency compared to many models invoked from literature for comparison. Following are the HFIT's properties: 1) Convergence ability of the evolutionary class algorithms (EA) or for that matter MOGP. 2) Approximation ability of the evolved hierarchical fuzzy system (tree model). 3) Convergence ability of DE in tree's parameters tuning. Subsequent discussions theoretically analyze each of these properties one-by-one.

A. Optimal tree structure through MOGP convergence

Evaluating the convergence of evolutionary class algorithms has been a challenging task because of their stochastic nature. Theoretical studies of EAs performed through various perspectives show that indeed an optimal solution is possible in a finite time. Initially, Goldberg and Sergret [67] showed convergence property of GA using a finite Markov chain analysis, where they considered GA with a finite population and recombination and mutation operators.

A different viewpoint of MOGP convergence (EAs in general) can be referred to as by using Banach fixpoint theorem described in [68]. Banach fixpoint theorem [69] states that on a metric space a constructive mapping f has a unique fixpoint, i.e., for an element x, $f(x) = x$. Therefore, Banach fixpoint theorem can explain MOGP convergence with only assumption that there should be an improvement of the population (not necessarily of the optimal solution) from one generation to another. Banach fixpoint theorem also indicates that if MOGP semantics is to be considered as a transformation between one population to another and if it is possible to obtain a metric space in which transformation is constructive, then MOGP converges to a optimal population W*, i.e., to a population containing optimal solution.

A mapping f defined on elements of ordered pair set S is constructive if the distance between $f(x)$ and $f(y)$ is less than x and y for any $x, y \in S$. Now, distance mapping $\delta : S \times S \to \mathbb{R}$ is a metric space iff for any $x, y \in S$ the following condition satisfy:

- $\delta(x, y) \geq 0$ and $\delta(x, y) = 0$ if $x = y$
- $\delta(x, y) = \delta(y, x)$
- $\delta(x, y) + \delta(y, z) \geq \delta(x, z)$

Let $\langle S, \delta \rangle$ be a metric space and $f : S \to S$ be a mapping, then f is *constructive* iff there is a constant $\epsilon \in [0, 1)$ such that for all $x, y \in S$

$$\delta(f(x), f(y)) \leq \epsilon\delta(x, y) \tag{30}$$

Therefore, for Banach theorem's formulation, the completeness of the metric space needs to be defined. Now, metric space elements $p_0, p_1, \ldots$ are a Cauchy sequence iff for any $\epsilon > 0$, there exist k such that for all $m, n > k$, $\delta(p_m, p_n) < \epsilon$. It also follows that, if such Cauchy sequence $p_0, p_1, \ldots$ has a limit $p = \lim_{n\to\infty} p_n$, then metric space is complete.

Theorem 1. *For a complete metric space $\langle S, \delta \rangle$ and constructive mapping $f : S \to S$, mapping f has a unique fixpoint $x \in S$ such that for any $x_0 \in S$*

$$x = \lim_{i\to\infty} f^i(x_0)$$

where $f^0(x_0) = x_0$ and $f^{i+1}(x_0) = f(f^i(x_0))$

Proof. A proof of Banach theorem can be found in [70, p. 60] described as method of successive approximation. □

In this article, it is necessary to show that if a metric space S for MOGP population can be obtained, then any constructive mapping f in MOGP will contain a unique fixpoint. The proposed MOGP has a fixed population size (say n), i.e., each population contain n individuals, and in each generation, the total fitness of the population is expected to increase. Let θ be a function that computes the fitness of a population, which is expressed as:

$$\theta(\mathrm{W}) = \frac{1}{n} \sum_{\mathbf{w}_i \in \mathrm{W}} \frac{\sum_{\mathbf{w}_i \in \mathrm{W}} E(\mathbf{w}_i)}{E(\mathbf{w}_i)} \tag{31}$$

where function $E(\mathbf{w}_i)$ evaluates RMSE of each $\mathbf{w}_i$. Now, distance mapping $\delta : S \times S \to \mathbb{R}$, where S is a set of MOGP populations, can be defined as:

$$\delta(\mathrm{W}_1, \mathrm{W}_2) = \begin{cases} 0, & \mathrm{W}_1 = \mathrm{W}_2 \\ |\theta(\mathrm{W}_1)| + |\theta(\mathrm{W}_2)|, & \mathrm{W}_1 \neq \mathrm{W}_2 \end{cases} \tag{32}$$

It follows that

- $\delta(\mathrm{W}_1, \mathrm{W}_2) \geq 0$ and $\delta(\mathrm{W}_1, \mathrm{W}_2) = 0$ if $\mathrm{W}_1 = \mathrm{W}_2$ holds for any population W_1 and W_2 in MOGP.
- $\delta(\mathrm{W}_1, \mathrm{W}_2) = \delta(\mathrm{W}_2, \mathrm{W}_1)$ is obvious and
- $\delta(\mathrm{W}_1, \mathrm{W}_2) + \delta(\mathrm{W}_2, \mathrm{W}_3) = |\theta(\mathrm{W}_1)| + |\theta(\mathrm{W}_2)| + |\theta(\mathrm{W}_2)| + |\theta(\mathrm{W}_3)| \geq |\theta(\mathrm{W}_1)| + |\theta(\mathrm{W}_3)| = \delta(\mathrm{W}_1, \mathrm{W}_3)$

Therefore, MOGP has a metric space $\langle S, \delta \rangle$. Now, it only remains to show that the MOGP follows a constructive mapping $f : S \to S$, i.e., in each subsequent generation of MOGP, an improvement is possible. Altenberg [71] showed that by maintaining genetic operators, such as selection, crossover, and mutation, the evolvability of genetic programming can be increased. Additionally, Altenberg [71] analyzed the probability of a population containing fitter individuals than the previous population and offered the subsequent proof. It was observed that even for a random crossover operation, genetic programming evolvability can be ensured. It is then necessary to say that, indeed an MOGP can produce fitter population than the previous ones.

Let's depart from MOGP operations descriptions to continue with Banach theorem since it is now known that MOGP offers constructive mapping $f : S \to S$, for which t-th iteration population offers constructive mapping. In other words, $\theta(\mathrm{W}^t) < \theta(\mathrm{W}^{t+1})$, i.e., mapping $f(\mathrm{W}^t) = \mathrm{W}^{t+1}$ holds. It follows that

$$\delta(f(\mathrm{W}_1^t), f(\mathrm{W}_2^t)) < \delta(\mathrm{W}_1^t, \mathrm{W}_2^t)$$

Moreover, it satisfies Banach fixpoint theorem. Hence,

$$\mathrm{W}^* = \lim_{i \to \infty} f^i(\mathrm{W}^0) \tag{33}$$

It indicates that MOGP converges to a population W^*, which is a unique fixpoint in a population space.

Remark 1. *It is evident from MOGP operation that it produces an optimal tree structure from a population space. Although obtaining optimality in the tree design using MOGP is sufficient to claim the formation of a function that can approximate to a high degree of accuracy, it is necessary to investigate the approximation capability of the hierarchical fuzzy system developed in the form of a tree structure.*

B. Approximation ability of hierarchical fuzzy inference tree

This Section describes the approximation capability of an HFIT, which is a result of MOGP operation. Theoretical studies of special cases of the hierarchical fuzzy systems are provided in [27], [32]. Whereas, the proposed HFIT produces a general hierarchical fuzzy system. In HFIT, not only a cascaded hierarchy of fuzzy system (a fuzzy subsystem takes input only from its previous fuzzy subsystem [27]) can be produced, but a general hierarchical fuzzy system, in which a fuzzy subsystem can take inputs from any previous layer fuzzy subsystem, can be produced. A hierarchical fuzzy system described in [30] resembles the hierarchical fuzzy system produced by HFIT. To show the approximation capability of the proposed HFIT, it requires coming to the conclusion that the proposed HFIT is analogous to the hierarchical fuzzy system described by Zeng and Keane [30].

Let's perform an analogy between the proposed HFIT and the concept of a natural hierarchical fuzzy system described by Zeng and Keane [30]. To show such an analogy, at first, it needs to establish the definition of the natural hierarchical structure of a continuous function, then it will be necessary to show that, for any such continuous function, a hierarchical fuzzy system exists.

Let's take the example of the HFIT shown in Fig. 3(a), which can be represented as natural hierarchical structure of a continuous function. The tree in Fig. 3(a) gives the output y_3 from node N_3. Moreover, the tree in Fig. 3(a) gives the following functions:

$$y_3 = N_3(y_1, y_2, x_3) \quad y_1 = N_1(x_1, x_2) \quad y_2 = N_2(x_4, x_5)$$

It can also be expressed as:

$$N(x_1, x_2, x_3, x_4, x_5) = N_3[N_1(x_1, x_2), N_2(x_4, x_5), x_3] \tag{34}$$

It follows that, for a given function $y = N(x_1, x_2, x_3, x_4, x_5)$, if there exist functions N_3, N_2, N_1 such that function (34) can be obtained, then function $N(x_1, x_2, x_3, x_4, x_5)$ can be represented as hierarchical structure.

For simplicity's sake, let's take the case of a two-stage tree, where the top layer node is denoted by N^1 and its output is by y. Similarly, second layer nodes are denoted by N_i^2 and their outputs are by y_i^2, for $1 \geq i \leq m$. Therefore, a natural hierarchical structure can be defined as:

Definition 1 (Natural Hierarchical Structure)**.** *Let $y = N(x_1, \ldots, x_n)$ be a multi-input-single-output continuous function with n input variables $\boldsymbol{x} = \langle x_1, \ldots, x_n \rangle$ defined on input space $U = \prod_{i=1}^{n} U_i \subset \mathbb{R}^n$ and the output y defined on the output space $V \subset \mathbb{R}$. If there exist $m + 1$ continuous functions*

$$\begin{aligned} y &= N^1(y_1^2, \ldots, y_m^2, x_i^1, \ldots, x_{d_1^i}^1) \\ y_j^2 &= N_j^2(x_i^{2j}, \ldots, x_{d_{2j}^i}^{2j}) \end{aligned} \tag{35}$$

and the functions have inputs $x_{d_1^i}^1$ and $x_{d_{2j}^i}^2$, where $d_1 < n$ and $d_{2j} < n$ are input dimensions at the top and second stage of hierarchy, respectively, such that

$$N(x_1, \ldots, x_n) = N^1[N_1^2(x_i^{21}, \ldots, x_{d_{21}^i}^{21}), \ldots, N_m^2(x_i^{2m}, \ldots, x_{d_{2m}^i}^{2m}), x_i^1, \ldots, x_{d_1^i}^1] \tag{36}$$

then $N(x_1, \ldots, x_n)$ is a continuous function with natural hierarchal structure.

Such form of natural hierarchical structure also possesses separable or arbitrarily separable hierarchical structural property, i.e., the individual functions can be decomposed [30]. Now, from Kolmogorov's Theorem [72], the following can be stated: Any continuous function $N(x_1, \ldots, x_n)$ on $U = \prod_{i=1}^{n}[\alpha_1, \beta_i]$ (α_i and β_i define the input range) can be represented as a sum of $2n + 1$ continuous functions with an arbitrarily separable hierarchical structure. This statement concludes to the following theorem.

Theorem 2. *Let $N(\boldsymbol{x})$ be any continuous function on $U = \prod_{i=1}^{n}[\alpha_i, \beta_i]$ and its hierarchical structure representation be $N(\boldsymbol{x}) = N^1[N^2(\boldsymbol{x}), \ldots, N^m(\boldsymbol{x})]$, in which $N^j(\boldsymbol{x})(j = 1, \ldots, m)$ are continuous functions with natural hierarchical structure, then for any given $\epsilon > 0$, there exists a hierarchical fuzzy system*

$$G(\boldsymbol{x}) = G^1[G^2(\boldsymbol{x}), \ldots, G^m(\boldsymbol{x})]$$

which has the same natural hierarchical structure as $N(\boldsymbol{x})$ *such that*

$$\|N - G\|_\infty < \epsilon \tag{37}$$

$$\|N^i - G^i\|_\infty < \epsilon \quad i = 0, 1, \ldots, m \tag{38}$$

and the same holds between the sub-functions of $N^i(\boldsymbol{x})$ *and the fuzzy subsystems of* $G^i(\boldsymbol{x})(i = 0, 1, \ldots, m)$.

Proof. Proof of Theorem 2 can be found in [30]. □

Remark 2. *It is to note that Theorem 2 shows that hierarchical structure of fuzzy systems is universal approximators. Therefore, they can approximate any continuous function* $N(\boldsymbol{x})$ *to any degree of accuracy as-well-as they can approximate each component of that function. Hence, the proposed HFIT that can form a natural hierarchical structure can achieve universal structure approximation.*

Another property of the proposed HFIT is the parameter tuning, which is performed by a global optimizer (e.g., DE was applied in this research). Hence, it is required to investigate the convergence ability of the DE algorithm in parameter tuning of HFIT.

C. Optimal parameter through differential evolution convergence

Convergence property and efficiency of DE is well studied [73], [74]. A probabilistic viewpoint of DE convergence followed by a description of global convergence condition for DE is described in [75]. They show that indeed DE converges to an optimal solution. Similarly, Zhang and Sanderson [74] studied the various property of DE, such as mutation, crossover and recombination operators that influence the DE convergence. DE follows a similar property as of EA class algorithms described in Section IV-A. Hence, its global convergence ability is not different than the one described for MOGP, and indeed it finds an optimal parameter vector for HFIT.

D. Comparative study of HFIT with other models

The proposed HFIT learns knowledge contained in the training data through adaptation in its structure and the rules generated at its nodes. Such a process of learning/acquiring knowledge

from data is somehow similar to the models having network-like layered architecture, i.e., ANFIS-like approaches, which usually have 4 or 5 or 6 layered network structure. However, HFIT's strength comes from its adaptive structure formation, whereas most of the network-like models have fixed layer structure.

1) Flexible structure formation: Specifically, the models depending on layered structure (e.g., HyFIS, DENFIS, D-FNN, EFuNN, FALCON, GNN, SaFIN, SONFIN, SuPFuNIS, eT2FIS, IT2FNN-SVR-N/F, McIT2FIS-UM/US, RIT2FNS-WB, SEIT2FNN, SIT2FNN, TSCIT2FNN, etc.) can only provide adaptation in the number of generated rules in their hidden layer by keeping the input (first) and output (last) layer fixed. Network-like model's fixed layered architecture in some sense limits their representational flexibility as compared to HFIT.

In Section IV-B, it was shown that HFIT has the capability of representing any continuous function in any natural and arbitrarily separable hierarchical form. Therefore, it can be said that the network-like models that grow rules only in one direction have a shortfall in structural representation compared to HFIT, which can grow in layer-wise as-well-as breadth-wise.

2) Diverse fuzzy rules formation: Additionally, the interaction of one RB to another through the structural representation is what sets HFIT apart from the other models, which generate only a single RB and do not have the interaction as it is in HFIT. Moreover, nodes in HFIT take difference input's combination govern by MOGP. Therefore, HFIT nodes exhibit heterogeneity, which drives the formation diverse rules in the nodes of HFIT. Whereas, rules in network-like models use same combination inputs while adding rules in the hidden layer during their training process.

3) Automatic fuzzy set selection: Adaptation in the most of the network-like models is due to the input space partitioning (usually for choosing the number of membership function at the second layer) in two or three fixed fuzzy sets or by using some clustering method, which directly influences the number of rules to form in the third layer (usually called rule layer). The necessity of predefining the number of clusters is basic disadvantage with the clustering based partitioning. Some of the practices in the clustering based partitioning, like the one in SaFIS, are devoted to improving the clustering algorithms to avoid the requirements of such predefinition. However, the overlapping of the membership function of the fuzzy sets is another common problem with clustering based input space partitioning [19]. In [76], authors pointed out four different cases of membership function's overlapping and proposed subsethood method to transmit the overlapping information to the rules layer.

On the other hand, HFIT does not use clustering to determine input space partitions. Instead, each input at each HFIT's node is partitioned into two fuzzy sets, which is eventually determined by MOGP through evolution. Section IV-A shows that MOGP finds an optimum solution through its iterations and the use of genetic operators. Hence, MOGP at it best avoids the overlapping of the membership function of the fuzzy sets and also eliminates the requirement of an external agent for input space partitioning.

4) Minimal feature set selection: Feature selection is another important aspect of HFIT. The network-like models such as EFuNN and DENFIS also does feature selection externally (say by external agents). However, in a sense, feature selection in such kind of models do not have direct participation in the structural representation of knowledge contained in training data. Whereas, feature selection is an integral part HFIT's learning process. Hence, feature selection performed by HFIT incorporates knowledge contained in training data into its structural representation in an explicit way compared to other network-like models. Since an external agent performs feature selection in the network-like models and many other models do not even perform feature selection, they are disadvantageous compared to HFIT when in comes to solving high-dimensional problems.

5) Parameter tuning: Finally, most of the models such as HyFIS, DENFIS, SaFIN, SONFIN, SEIT2FNN, McIT2FIS-UM/US, etc. employ gradient-based methods (e.g., backpropagation) for the parameter tuning. The gradient-based techniques are known as local optimizers, which lacks exploration capability compared to global optimizers (e.g., DE) [77]. HFIT employs DE for its parameter optimization. When it comes to comparing models theoretically, it is not necessary to go deep in parameter tuning debate since such parameter tuning method like DE can also be applied to other models and backpropagation can be applied to HFIT. However, at present scenario, a combined effort of the proposed HFIT model, in this article, have an advantage compared to other models.

V. Empirical Evaluation

This section describes the evaluated results of the proposed algorithms T1HFIT$^{\mathrm{S}}$, T1HFIT$^{\mathrm{M}}$, T2HFIT$^{\mathrm{S}}$, and T2HFIT$^{\mathrm{M}}$ on six example problems. Assume that the datasets in the examples are of the form: $(\mathbf{X}, \mathbf{d})$, where $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ is the set of the input vectors and $\mathbf{d} = \langle d_1, d_2, \ldots, d_N \rangle$ is the desired output vector. Here, the dataset has N input–output patterns (pairs) and if the vector $\mathbf{y} = \langle y_1, y_2, \ldots, y_N \rangle$ is the predicted output vector, then the performance of

an algorithm for the dataset $(\mathbf{X}, \mathbf{d})$ can be measured using RMSE E as defined in (26) and correlation coefficient r between the desired output vector $\mathbf{d}$ and $\mathbf{y}$ as:

$$r = \frac{\sum_{i=1}^{N} \left(d_i - \bar{d}\right)\left(y_i - \bar{y}\right)}{\sqrt{\sum_{i=1}^{N} \left(d_i - \bar{d}\right)^2 \sum_{i=1}^{N} \left(y_i - \bar{y}\right)^2}} \tag{39}$$

where $\bar{d}$ and $\bar{y}$ are the means of the vectors $\mathbf{d}$ and $\mathbf{y}$. For simplicity's sake, the training and the test RMSEs were represented as E_n and E_t, respectively. Similarly, the training and the test correlation coefficients were represented as r_n and r_t, respectively. Additionally, the model's complexity $c(\mathbf{w})$ and training time (in minutes) were reported. The reported training time included the time taken to create a tree structure, tune the tree parameters, partition the dataset (file input–output operations), write the developed model to a file, display the tree on a GUI, and compute RMSE and correlation coefficient. The parameter setting mentioned in Table I was used to train the proposed algorithms, which was developed as a software tool and is available at http://dap.vsb.cz/sw/hfit/. The experiments were conducted on a Windows Server R2 that had a 20 core and 700 GB RAM. Each run of experiments was conducted with the random seeds generated from the system. The proposed algorithms were compared with the algorithms collected from the literature (Table II).

TABLE I

PARAMETER SETTING FOR THE EXPERIMENTS

Algorithm training parameter	Value
Maximum depth (layers) of a tree	4
Maximum inputs to an FIS node	4
Membership function search range	[0,1]
GP population	50
CP mutation probability pm	0.2
GP crossover probability $pc = 1 - pm$	0.8
GP mating pool size	25
GP tournaments selection size	2
GP iterations	500
DE population	50
DE mutation factor F	0.7
DE crossover factor cr	0.9
DE iterations	5000

TABLE II

DESCRIPTIONS OF THE EXISTING FIS ALGORITHMS ADOPTED FOR THE PERFORMANCE COMPARISONS

FIS	Algorithm	Ref.	Description	Type	Parameter tuning
Type–1	DENFIS	[18]	Dynamic evolving neural-fuzzy inference system	TSK	Least-square estimator
	D-FNN	[10]	Dynamic fuzzy neural networks	TSK	Backpropagation algorithm
	EFuNN	[78]	Evolving fuzzy neural networks	Mamdani	Widrow–Hoff least square
	FALCON	[79]	ART-based fuzzy adaptive learning control network	$--$	Backpropagation algorithm
	GNN	[80]	Granular neural networks	$--$	Genetic algorithm
	H-TS-FS	[35]	Hierarchical Tukagi–Sugno fuzzy system	TSK	Evolutionary programming
	HyFIS	[81]	Hybrid neural fuzzy inference system	$--$	Gradient descent learning
	IFRS and AFRS	[82]	Incremental and aggregated fuzzy relational systems	Mamdani	Backpropagation algorithm
	RBF-AFA	[83]	Radial basis function based adaptive fuzzy systems	TSK	Gradient descent learning
	SaFIN	[19]	Self-adaptive fuzzy inference network	Mamdani	Levenberg-Marquardt method
	SONFIN	[15]	Self-constructing neural fuzzy inference network	TSK	Backpropagation algorithm
	SuPFuNIS	[76]	Subsethood-product fuzzy neural inference system	$--$	Gradient descent learning
	SVR-FM	[84]	Support-vector regression fuzzy model	TSK	Support vector regression
Type–2	eT2FIS	[85]	Evolving type-2 neural fuzzy inference system	Mamdani	Gradient descent learning
	IT2FNN-SVR-N/F	[86]	IT2fuzzy-NN-support-vector regression-fuzzy and numeric	TSK	Support vector regression
	McIT2FIS-UM/US	[24]	Metacognitive interval type-2 neuro-fuzzy inference system	TSK	Gradient descent learning
	NNT1FW and NNT2FW	[87]	Type-1 and type-2 fuzzy backpropagation neural networks	TSK	Backpropagation algorithm
	RIT2FNS-WB	[17]	Reduced IT2NFS-weighted bound-set	TSK	Gradient descent learning
	MRIT2NFS	[17]	Reduced IT2NFS-weighted bound-set	Mamdani	Gradient descent learning
	SEIT2FNN	[16]	Self-evolving IT2FIS	TSK	Kalman filtering algorithm
	SIT2FNN	[22]	Simplified Interval Type-2 Fuzzy Neural Networks	TSK	gradient descent learning
	T2FLS	[88]	Interval type-2 fuzzy logic system (TSK and singleton)	TSK	$--$
	T2FLS-G	[89]	Gradient-descent based IT2FIS tuning	TSK	Derivation-based learning
	TSCIT2FNN	[21]	Compensatory interval type-2 fuzzy neural network	TSK	Kalman filter algorithm

A. Example 1—System Identification

Online identification of the nonlinear system is a widely studied problem. The significance of this problem is evident from its usage in the literature for the validation of the approximation algorithms [16], [21], [24], [86], [90]. The nonlinear system identification of the plant is described by the following nonlinear difference equation:

$$y_p(k+1) = \frac{y_p(k)}{1+y_p(k)^2} + u^3(k) \tag{40}$$

where $[u(k), y_p(k)]$ is the input–output pair of the single input and the single output plant at the time k and $y_p(k+1)$ is the one step ahead prediction. Hence, the objective is to predict $y_p(k+1)$ of the system based on the sinusoidal input $u(k) = \sin(2\pi k/100)$ and the current output $y_p(k)$. Let us assign the input $x_1 = u(k)$ and the input $x_2 = y(k)$.

The training patterns were generated with $k = 1, \ldots, 200$ and $y_p(1) = 0$. Similarly, the test patterns were generated for $k = 201, \ldots, 400$ as mentioned in [17]. Therefore, for the training, the inputs were $u(k)$ and $y_p(k)$, and the desired output was $y_p(k+1)$. The training and test were repeated ten times. Such repetitions were performed mainly for assessing an average performance of the proposed algorithms, which is shown in Table III(a). Since the experiments were repeated ten times, ten different models were obtained for each algorithm. The results of the best models (regarding their RMSEs) were compared with the best results available in the literature (Table III(b)).

The performance statistics, shown in Table III(a), are evidence of the efficiency of the proposed algorithms. They show that the mean correlation coefficients r_n and r_t of training and test sets are 1.00 and 1.00, respectively, which indicate that the algorithm consistently performed with a high accuracy. Moreover, such consistency of high accuracy is evident from the obtained small standard deviations (STD) of the training and test RMSEs and correlation coefficients (Table III(a)).

Interestingly, the Pareto-based multiobjective training offered less complex models (the mean parameter count $c(\mathbf{w})$ of T1HFIT$^{\text{M}}$ was 34.4 compared to 57.2 of T1HFIT$^{\text{S}}$ and $c(\mathbf{w})$ of T2HFIT$^{\text{M}}$ was 90.4 compared to 152.0 of T1HFIT$^{\text{S}}$) with high accuracies (Table III(a)). Additionally, the training time taken by T1HFIT$^{\text{M}}$ and T2HFIT$^{\text{M}}$ was much less than by T1HFIT$^{\text{S}}$ and T2HFIT$^{\text{S}}$. Hence, the Pareto-based multiobjective was advantageous to use, which provided the option of choosing the best solution from a Pareto-front. An example of a Pareto-front is shown in Fig. 5.

For the performance comparisons, the SaFIN result was collected from [19], and FALCON and SONFIN from [16]. The results of T2FLS (singleton) and T2FLS (TSK) were obtained from [16]; FT2FNN, TSCIT2FNN, T2TSKFNS, and T2FNN from [21]; SEIT2FNN, MRI2NFS, RIT2NFS-WB, and T2FLS-G from [17]; and SIT2FNN from [22]. Table II contains a detailed description of these algorithms.

Two parameters may be used for comparing the algorithms: 1) the training and test RMSEs and 2) the parameter count $c(\mathbf{w})$. From the performance comparisons shown in Table III(b), it was found that the proposed algorithms T1HFIT$^{\text{S}}$ and T1HFIT$^{\text{M}}$ were better than the T1FIS algorithms FALCON, SaFIN, and SONFIN. SONFIN offered the test RMSE $E_t = 0.0085$ with the smallest parameter count $c(\mathbf{w}) = 36$; whereas, the proposed algorithm T1HFIT$^{\text{M}}$ offered the better test RMSE $E_t = 0.0041$ with a slightly larger parameter count $c(\mathbf{w}) = 40$.

TABLE III

PERFORMANCE EVALUATION ON SYSTEM IDENTIFICATION (EXAMPLE-1)

(a) Performance Statistics (10 repetitions)

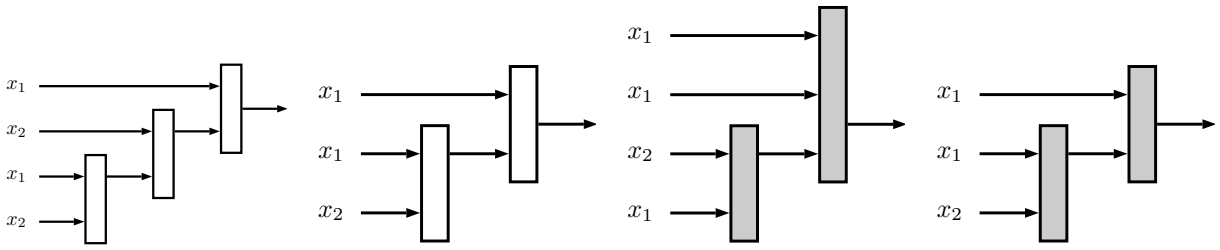
		T1HFIT$^{\text{S}}$	T1HFIT$^{\text{M}}$	T2HFIT$^{\text{S}}$	T2HFIT$^{\text{M}}$
E_n	Best	0.0043	0.0041	0.0033	0.0028
	Mean	0.0181	0.0257	0.0123	0.0184
	STD	0.0167	0.0164	0.0074	0.0105
r_n	Best	1.00	1.00	1.00	1.00
	Mean	1.000	0.999	1.000	1.000
	STD	0.0006	0.0007	0.0001	0.0002
E_t	Best	0.0020	0.0041	0.0034	0.0028
	Mean	0.0169	0.0262	0.0125	0.0187
	STD	0.0173	0.0171	0.0076	0.0109
r_t	Best	1.00	1.00	1.00	1.00
	Mean	1.000	0.999	1.000	1.000
	STD	0.0006	0.0007	0.0001	0.0002
$c(\mathbf{w})$	Best	20	20	72	36
	Mean	57.2	34.4	152	90.4
Time	Best	3.21	1.52	7.82	3.23
	Mean	6.27	2.91	8.91	5.14

(b) Performance Comparison

	Algorithm	E_n	E_t	$c(\mathbf{w})$
Type–1	FALCON	0.0200		54
	SaFIN		0.0120	
	SONFIN	0.0080	0.0085	36
	T1HFIT$^{\text{S}}$	0.0043	0.0043	60
	T1HFIT$^{\text{M}}$	0.0041	0.0041	40
Type–2	T2FLS (singleton)	0.0306	–	120
	FT2FNN	0.0388	–	36
	T2FLS (TSK)	0.0217	–	120
	TSCIT2FNN	0.0080	–	34
	T2TSKFNS	–	0.0324	24
	T2FNN	–	0.0281	36
	SIT2FNN	–	0.0241	36
	RIT2NFS-WB	0.0073	0.0151	24
	MRI2NFS	0.0042	0.0051	36
	T2FLS-G	0.0214	0.0379	36
	SEIT2FNN	0.0022	0.0022	84
	T2HFIT$^{\text{S}}$	0.0033	0.0034	118
	T2HFIT$^{\text{M}}$	0.0028	0.0028	72

Similarly, the proposed T2FIS algorithms T2HFIT$^{\text{S}}$ and T2HFIT$^{\text{M}}$ offered better performance compared to the algorithms T2FLS (Singleton), T2FLS (TSK), TSCIT2FNN, T2TSKFNS, T2FNN, SIT2FNN, RIT2NFS-WB, MRI2NFS. The algorithm SEIT2FNN reported test RMSE $E_t = 0.0022$ and the parameter count was 84; whereas, in comparison to SEIT2FNN, the algorithm T2HFIT$^{\text{M}}$ offered a slightly higher test RMSE $E_t = 0.0028$, but had a lower parameter count $c(\mathbf{w})$, i.e., 72.

The time comparison, however, is limited since the training time depends on several factors: 1) the type of programming language used; 2) the platform and its configurations on which programs were executed; 3) the way data were fed for the training; 3) the status of the cache memory (in the case CPU time is observed); etc. It may be noted that, from the available training time reported in the literature, the MRI2NFS, RIT2NFS-W, T2FLS-G, SEIT2FNN approximately takes 0.15, 0.17, 2.41, and 2.24 minutes (CPU time only) respectively. This is in comparison



(a) T1HFIT$^{\mathrm{S}}$: $E_n = 0.0043$ (b) T1HFIT$^{\mathrm{M}}$: $E_n = 0.0041$ (c) T2HFIT$^{\mathrm{S}}$: $E_n = 0.0034$ (d) T2HFIT$^{\mathrm{M}}$: $E_n = 0.0028$

Fig. 6. Example–1: designed HFIT, where the shaded nodes indicate T2FIS.

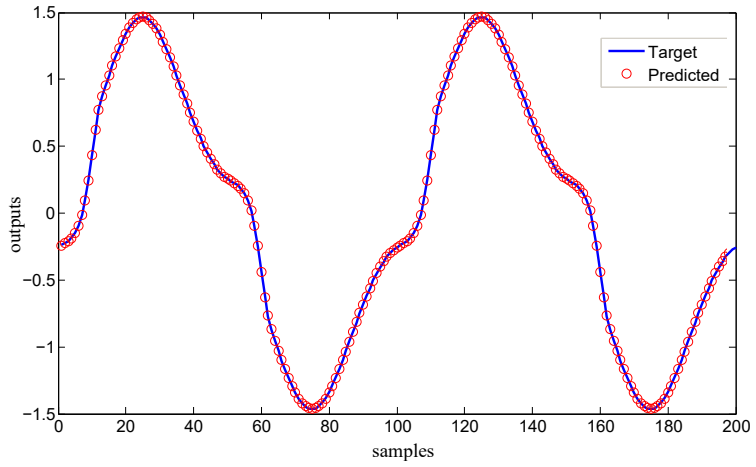


Fig. 7. Example–1: target versus predicted test values. The test outputs belong to algorithm T2HFIT$^{\mathrm{M}}$, which has the test RMSE $E_t = 0.0028$.

to T1HFIT$^{\mathrm{M}}$ and T2HFIT$^{\mathrm{M}}$, which take 1.52 and 3.23 minutes (including CPU time, other file operations, etc.) respectively. Since the training times taken by the algorithms were close to one another and the time comparison has limitations, it may be concluded that the proposed models performed efficiently. This is also evident from the performance statistics given in Table III(a) and the performance comparison is provided in Table III(b).

The best models obtained using the proposed algorithms are illustrated in Fig.6, which shows the hierarchical structure of the derived models and the selected inputs are indicated by x_i in the models. The rectangular blocks in Fig 6 show the nodes (a T1FIS or T2FIS) of the tree (hierarchical structure). The target and predicted value plots of 200 samples are shown in Fig. 7.

B. Example 2—Noisy Chaotic Time Series Prediction

1) Case–Clean Set: A chaotic time series dataset, the Mackey-Glass chaotic time series, was used in this example, which was generated using the following delay differential equation:

$$\frac{dx(k)}{dk} = \frac{0.2x(k-\tau)}{1+x^{10}(k-\tau)} - 0.1x(k) \tag{41}$$

where delay constant $\tau > 17$ and k is the time step. In this example, the objective was to predict $x(k)$ using the past outputs of the time series as mentioned in [86]. Hence, the input–output pattern was of the form:

$$[x(k-24), x(k-18), x(k-12), x(k-6); x(k)]\,.$$

Let us say that the inputs are $x_1 = x(k-24)$, $x_2 = x(k-18)$, $x_3 = x(k-12)$, and $x_4 = x(k-6)$. For the training of the proposed algorithms, a total of 1000 patterns were generated from $k = 124$ to 1123, with the parameter τ being set to 30 and $x(0)$ being set to 1.2 [86]. This set of training patterns were clean (no noise was added). From the generated clean patterns, as mentioned in [86], the first 500 patterns (clean training set) were used for training purposes and the second 500 patterns (clean test set) were used for test purposes. Aiming to assess the average performance of the proposed algorithms, ten repetitions of training and testing were performed using clean training and test sets, and the results were collected accordingly (Table IV(a)). Table IV(b) shows the comparison of results of the proposed algorithms (the best among ten models) with the best results reported by the algorithms listed in Table II.

For this example (clean set), the performance statistics are shown in Table IV(a). The obtained statistics illustrate that the proposed algorithms T1HFIT$^{\text{S}}$, T1HFIT$^{\text{M}}$, T2HFIT$^{\text{S}}$, and T2HFIT$^{\text{M}}$ performed with high accuracies. It shows that the mean correlation coefficient r_n of the training set of all algorithms is 1.00, and the mean correlation coefficient r_t of the test set of the algorithms T1HFIT$^{\text{S}}$, T1HFIT$^{\text{M}}$, T2HFIT$^{\text{S}}$, and T2HFIT$^{\text{M}}$ are 0.9858, 0.9864, 0.9783, and 0.9912 respectively. That is the test correlation coefficients are closer to 1.00 (a high positive correlation between target and predicted outputs). Such performance indicates that the algorithms consistently performed with a high accuracy, and the obtained low values of STDs are evidence of this fact (Table IV(a)).

Moreover, the Pareto-based multiobjective training offered less complex models (the mean parameter count $c(\mathbf{w})$ of T1HFIT$^{\text{M}}$ was 57.6 compared to 71.6 of T1HFIT$^{\text{S}}$ and the $c(\mathbf{w})$ of T2HFIT$^{\text{M}}$ was 129.5 compared to 203.4 of T1HFIT$^{\text{S}}$) with high accuracies (Table IV(a)). Hence, like in example 1, in this example also the Pareto-based multiobjective was advantageous to use,

TABLE IV

PERFORMANCE EVALUATION ON CLEAN SET OF NOISY CHAOTIC TIME SERIES PREDICTION (EXAMPLE-2)

(a) Performance Statistics (10 repetitions)

		T1HFIT^{S}	T1HFIT^{M}	T2HFIT^{S}	T2HFIT^{M}
E_n	Best	0.0115	0.0115	0.0108	0.0032
	Mean	0.0345	0.0338	0.0413	0.0224
	STD	0.0163	0.0207	0.0221	0.0203
r_t	Best	1.00	1.00	1.00	1.00
	Mean	0.9858	0.9864	0.9783	0.9912
	STD	0.0117	0.0107	0.0182	0.0154
E_t	Best	0.0122	0.0119	0.0086	0.0058
	Mean	0.0414	0.0356	0.0427	0.0275
	STD	0.0224	0.0173	0.0234	0.0207
r_t	Best	1.00	1.00	1.00	1.00
	Mean	0.9786	0.9850	0.9769	0.9888
	STD	0.0211	0.0120	0.0195	0.0158
$c(\mathbf{w})$	Best	20	40	72	36
	Mean	71.6	57.6	203.4	129.5
Time	Best	8.42	5.51	21.33	7.91
	Mean	71.6	11.03	31.83	16.58

(b) Performance Comparison

	Algorithm	E_n	E_t	$c(\mathbf{w})$
Type–1	NNT1FW	–	0.0550	–
	AFRS	0.0267	0.0256	78
	IFRS	0.0240	0.0253	58
	H-TS-FS^{1}	0.0120	0.0129	148
	H-TS-FS^{2}	0.0145	0.0151	46
	RBF-AFA	–	0.0128	–
	HyFIS	–	0.0100	–
	D-FNN	–	0.0080	70*
	SuPFuNIS	–	0.0057	70*
	T1HFIT^{S}	0.0115	0.0122	60
	T1HFIT^{M}	0.0115	0.0119	40
Type–2	T2FLS (singleton)	–	0.0426	–
	T2FLS (TSK)	–	0.0431	–
	NNT2FW	–	0.0390	–
	SEIT2FNN^{1}	–	0.0034	126*
	SEIT2FNN^{2}	–	0.0053	90*
	T2HFIT^{S}	0.0108	0.0086	108
	T2HFIT^{M}	0.0032	0.0058	118

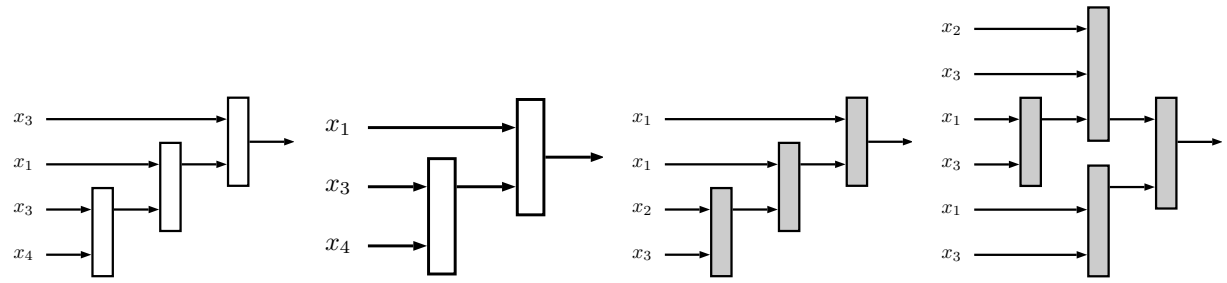
*This is approximately calculated. It may be larger.

which provided the option to choose the best solution from a Pareto-front. Fig. 5 illustrates a Pareto-front created during the multiobjective training of HFIT.

Table IV(b) describes the comparison between several algorithms on clean training and test set. The results of IFRS, AFRS, H-TS-FS1, and H-TS-FS2 were collected from [35]; RBF-AFA, HyFIS, D-FNN, and SuPFuNIS from [16]; NNT1FW and NNT2FW from [87]; and T2FLS (Singleton), T2FLS (TSK), and SEIT2FN from [16].

The training and test RMSEs and the parameter count $c(\mathbf{w})$ were used for comparing the algorithms, which is shown in Table IV(b). A training time comparison for this example cannot be performed because of the unavailability of the training time of other algorithms in the literature.

In T1FIS comparisons, it was found that the proposed algorithms T1HFIT^{S} and T1HFIT^{M} performed better than or were competitive with the algorithms NNT1FW, AFRS, IFRS, H-



(a) T1HFIT$^{\mathrm{S}}$: $E_n = 0.0115$ (b) T1HFIT$^{\mathrm{M}}$: $E_n = 0.0115$ (c) T2HFIT$^{\mathrm{S}}$: $E_n = 0.0108$ (d) T2HFIT$^{\mathrm{M}}$: $E_n = 0.0032$

Fig. 8. Example–2 Clean set: designed HFIT. The shaded nodes are T2FIS.

TS-FS, RBF-AFA, and HyFIS. The algorithms D-FNN and SuPFuNIS had better test RMESs $E_t = 0.008$ and $E_t = 0.005$, but their parameter counts were larger since the number of rules in each case was 10. Since each T1FS MF has at least two parameters and each rule has three free parameters at the consequent part, the number of parameter count for two input variables stands to at least 70 (this is an approximate calculation since D-FNN and SuPFuNIS may have other parameters that may increase the parameter count value). Whereas, the algorithms T1HFIT$^{\mathrm{S}}$ and T1HFIT$^{\mathrm{M}}$ had parameter counts equal to 60 and 40, respectively. Therefore, T1HFIT$^{\mathrm{S}}$ and T1HFIT$^{\mathrm{M}}$ are competitive with D-FNN and SuPFuNIS.

In T2FIS, the proposed algorithms clearly performed better than T2FLS (Singleton), T2FLS (TSK), and NNT2FW. Whereas, the performance of the proposed algorithms were competitive with SEIT2FNN^1 (without fuzzy set reduction) and SEIT2FNN^2 (with fuzzy set reduction) whose test RMSEs E_t were 0.0034 and 0.0058, respectively. The algorithm SEIT2FNN^1 had 28 fuzzy sets and SEIT2FNN^2 had 16 fuzzy sets (reduced), and each of these had seven rules. Hence, the parameter count of these algorithms stands to at least 126 and 90, respectively. On the other hand, the proposed algorithm T2HFIT$^{\mathrm{S}}$ had a test RMSE of $E_t = 0.0086$ (slightly larger than SEIT2FNN^1 and SEIT2FNN^2), but the parameter count was 108, which is smaller than SEIT2FNN^1. Similarly, the proposed algorithm T2HFIT$^{\mathrm{M}}$ had a test RMSE of $E_t = 0.0058$, which is close to SEIT2FNN^2 and the parameter count was smaller than SEIT2FNN^1 and closer to SEIT2FNN^2. Therefore, in this case, the proposed T2HFIT$^{\mathrm{M}}$ is as efficient as SEIT2FNN^1 and SEIT2FNN^2 are. Fig.8 shows the hierarchical structure of the best models obtained by the proposed algorithms. Additionally, the target versus prediction plot of test data samples is illustrated in Fig. 9.

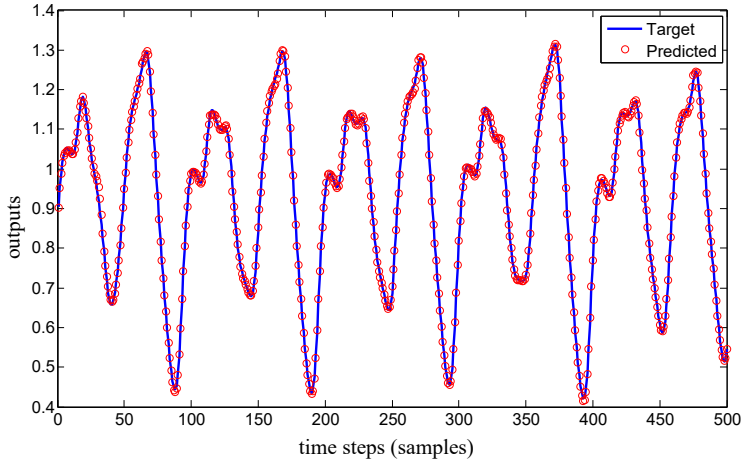


Fig. 9. Example–2 Clean set: target versus predicted test values. The test outputs belong to algorithm T2HFIT^{M} that has the test RMSE $E_t = 0.0058$.

2) Case–Noisy Set: The performances of the proposed algorithms were further evaluated for noisy patterns. Therefore, three training sets and three test sets were generated by adding Gaussian noise with a mean of 0 and STDs of 0.1, 0.2, and 0.3 to the original data $x(k)$ as described in [86]. These noisy training sets (with STDs 0.1, 0.2, and 0.3) were presented for the training of the proposed algorithms. With each training set of STDs of 0.1, 0.2, and 0.3, three test sets were given for testing: clean, STD 0.1, and STD 0.3. The obtained results were compared with the results reported in the literature (Table V).

Table V describes the comparisons between the results of the algorithms, where the results of SONFIN and SVR-FM were collected from [86], DENFIS and EFuNN from [85], SEIT2FNN, T2FLS-G, IT2FNN-SVR(N), IT2FNN-SVR(F) from [86], and eT2FIS from [22]. It is evident from the comparison of the results that the proposed algorithms performed efficiently over the noisy datasets and the obtained models were less complex than the other models listed in Table V. Particularly when T1FISs were compared. Moreover, for each noisy data (STD 0.1, STD 0.2, and STD 0.3), the proposed algorithms had a smaller parameter count and had a lower or competitive training RMSE E_n compared to other listed algorithms. In T1FIS comparisons, the SONFIN had a slightly better RMSE, but the number of parameters counts was larger than the proposed algorithms T1HFIT^{S} and T1HFIT^{M}. Similarly, in T2FIS comparison, the algorithm eT2FIS had slightly better RMSE than the other listed algorithms, but the models obtained using the proposed algorithms were less complex, i.e., had a smaller parameter count.

TABLE V

EXAMPLE 2–NOISY SET: PERFORMANCE COMPARISON

		Train	Test				Train	Test				Train	Test			
FIS	Algorithm	0.1	clean	0.1	0.3	$c(\mathbf{w})$	0.2	clean	0.1	0.3	$c(\mathbf{w})$	0.3	clean	0.1	0.3	$c(\mathbf{w})$
Type–1	SVR-FM	0.128	0.045	0.087	0.200	1127	0.229	0.089	0.109	0.189	1127	0.332	0.138	0.147	0.198	1127
	EFuNN	0.126	–	–	–	–	0.252	–	–	–	–	0.366	–	–	–	–
	DENFIS	0.116	–	–	–	–	0.214	–	–	–	–	0.306	–	–	–	–
	SONFIN	0.113	0.054	0.108	0.256	130	0.226	0.116	0.138	0.280	130	0.302	0.195	0.208	0.305	130
	T1HFIT^{S}	0.127	0.050	0.140	0.363	60	0.234	0.111	0.153	0.349	104	0.305	0.100	0.159	0.356	64
	T1HFIT^{M}	0.128	0.042	0.138	0.357	40	0.225	0.085	0.145	0.360	84	0.307	0.119	0.162	0.351	60
Type–2	T2FLS-G	0.133	0.074	0.103	0.220	110	0.238	0.125	0.132	0.200	110	0.357	0.232	0.234	0.264	110
	IT2FNN-SVR(N)	0.128	0.048	0.087	0.193	103	0.234	0.085	0.105	0.186	103	0.349	0.127	0.138	0.188	103
	IT2FNN-SVR(F)	0.127	0.046	0.088	0.215	103	0.233	0.083	0.103	0.180	103	0.347	0.121	0.131	0.184	103
	SEIT2FNN	0.123	0.049	0.097	0.212	110	0.225	0.083	0.113	0.228	110	0.319	0.196	0.197	0.254	110
	eT2FIS	0.120	0.059	0.107	0.214	–	0.225	0.083	0.132	0.247	–	0.327	0.102	0.152	0.278	–
	T2HFIT^{S}	0.128	0.039	0.135	0.355	108	0.227	0.079	0.143	0.348	82	0.314	0.100	0.148	0.354	144
	T2HFIT^{M}	0.123	0.042	0.135	0.365	72	0.233	0.087	0.144	0.348	72	0.311	0.097	0.148	0.356	108

C. Example 3—Miles-Per-Gallon Prediction Problem

To evaluate the performance of the proposed algorithms, a real-world MPG problem was used. The objective of this example was to predict or estimate the city-cycle fuel consumption in MPG. The MPG dataset was collected from the UCI machine learning repository [91]. This dataset has 392 samples, each of which has six input variables, but in this example, as mentioned in [24], three variables (x_1 = weight, x_2 = acceleration, and x_3 = model year) were selected. In the training process, 50% (196 patterns) of samples were randomly selected for training and the rest of the 50% (196 patterns) of samples were taken for testing. Such a process for the training set and test set selection was repeated ten times. Accordingly, the collected performance statistics are shown in Table VI(a).

The performances of the proposed algorithms were compared with the literature (Table VI(a)). However, the algorithms chosen from the literature were tested over fewer test samples. Therefore, the comparison shown in Table VI(a) is limited to the comparison of the training RMSE because all the mentioned algorithms were trained over the same number of training samples. For the comparisons, the T1FLS result was collected from [17] and the results of SEIT2FNN, RIT2NFS-WB, McIT2FIS-UM, and McIT2FIS-US were collected from [24].

TABLE VI

PERFORMANCE EVALUATION ON MILES-PER-GALLON PREDICTION PROBLEM (EXAMPLE-3)

(a) Performance Statistics (10 repetitions)

		T1HFIT^{S}	T1HFIT^{M}	T2HFIT^{S}	T2HFIT^{M}
E_n	Best	1.8931	2.2686	2.0881	1.9582
	Mean	2.7115	2.6037	2.4699	2.4052
	STD	0.5144	0.4071	0.4461	0.3774
r_n	Best	0.97	0.96	0.96	0.96
	Mean	0.921	0.941	0.946	0.950
	STD	0.1035	0.0218	0.0244	0.0160
E_t	Best	2.7550	2.7907	2.8383	2.6623
	Mean	4.2333	3.3349	3.4006	3.3172
	STD	0.5024	0.5720	0.7423	0.6855
r_t	Best	0.97	0.96	0.96	0.96
	Mean	0.921	0.941	0.946	0.950
	STD	0.1035	0.0218	0.0244	0.0160
$c(\mathbf{w})$	Best	20	20	108	118
	Mean	132	78.8	224	207.4
Time	Best	1.89	1.75	8.75	8.53
	Mean	12.04	6.75	14.91	12.33

(b) Performance Comparison (10 Repetitions)

	Algorithm	Mean E_n	STD	Mean E_t	STD	Samples (train, test)
Type–1	T1FLS	–	–	3.5960	–	196, 120
	T1HFIT^{S}	2.7115	0.5144	4.2333	0.5024	196, 196
	T1HFIT^{M}	2.6037	0.4071	3.3349	0.5720	196, 196
Type–2	McIT2FIS-US	2.7358	–	2.6770	–	196, 120
	SEIT2FNN	2.7161	–	2.7895	–	196, 120
	McIT2FIS-UM	2.6524	–	2.6486	–	196, 120
	RIT2NFS-WB	2.3685	–	2.7807	–	196, 120
	T2HFIT^{S}	2.4699	0.4461	3.4006	0.7423	196, 196
	T2HFIT^{M}	2.4052	0.3774	3.3172	0.6855	196, 196

The comparisons of the models in this example were based on the mean training and test RMSEs E_n and E_t obtained for the ten repetitions. However, the comparison on test RMSEs was limited since only 120 samples were used for testing by the algorithms considered from literature. Whereas, the algorithms proposed in this work used 196 samples for testing (Table VI(b)). It was observed that the proposed algorithms T2HFIT^{S} and T2HFIT^{M} outperformed all the other algorithms except for RIT2NFS-WB, which had a slightly better training RMSE $E_n = 2.3685$ in comparison to the training RMSEs $E_n = 2.4699$ and $E_n = 2.4052$ of T2HFIT^{S} and T2HFIT^{M}, respectively. Since the performance comparisons were based on the average value of ten repetitions, the model's hierarchical structures are not presented for this example.

From the available training time reported in the literature, it may be noted that the algorithms McIT2FIS-UM, McIT2FIS-US, RIT2NFS-WB, and SEIT2FNN take 0.0025, 0.003, 0.16, and 0.33 minutes (CPU time only) compared to T2HFIT^{M}, which takes 8.23 minutes. However, it may be noted that T2HFIT^{M} is a two-phase population-based learning algorithm, whereas the

other algorithms are single-solution based algorithms.

D. Example 4—Abalone Age Prediction

In this example, a prediction problem was taken in which a person's age was predicted based on their physical measurements. The Abalone dataset was collected from the UCI machine learning repository [91]. It has 4177 data samples, each of which has seven input variables (x_1 = length, x_2 = diameter, x_3 = height, x_4 = whole weight, x_5 = shucked weight, x_6 = viscera weight, and x_7 = shell weight) and one output variable (rings). To train the proposed algorithms, 80% (3342 patterns) of samples were randomly taken for training and 20% (835 patterns) remaining samples were taken for testing. Additionally, in this work, to assess the average performance of proposed algorithms, training process was repeated ten times, and the collected results are summarized in Table VII(a).

The obtained results are compared with the results reported in the literature (Table VII(b)). For the comparisons, the results of General, HS, CCL, and Chen&Cheng were collected from [17], and the results of SEIT2FNN, RIT2NFS-WB, McIT2FIS-UM, and McIT2FIS-US were collected from [24]. The algorithms General [92], CCL [93], HS [94], and WFRI-GA [95] were fuzzy interpolate reasoning methods, where WFRI-GA was based on the genetic algorithm and the algorithm 'General' implemented the Mamdani type FIS. It is evident from the results in Table VII(b) that the proposed algorithms (both T1FIS and T2FIS) outperformed the algorithms considered for comparisons.

However, when comparing the test RMSEs, McIT2FIS-US, McIT2FIS-UM, and RIT2NFS-WB had a slight edge over T2HFIT$^{\mathrm{S}}$ and T2HFIT$^{\mathrm{M}}$, but the parameter count of T2HFIT$^{\mathrm{M}}$ was smallest among all the algorithms, and it had the lowest training error. Hence, it may be concluded that T2HFIT$^{\mathrm{M}}$ is the best performing algorithm for example 4. T2HFIT$^{\mathrm{M}}$ performance falls behind only in training time comparison because T2HFIT$^{\mathrm{M}}$, being a population based algorithm, takes longer training time than the other algorithms. The algorithms McIT2FIS-US, McIT2FIS-UM, RIT2NFS-WB, and SEIT2FNN take 1.81, 2.35, 5.48, and 17.33 minutes (CPU time only) compared to T2HFIT$^{\mathrm{M}}$, which takes 65.02 minutes. It is important to note that the other algorithms are single solution based algorithms. The best-performing models of the proposed algorithms are illustrated in Fig. 10, where the selected input feature is indicated by x_i.

TABLE VII

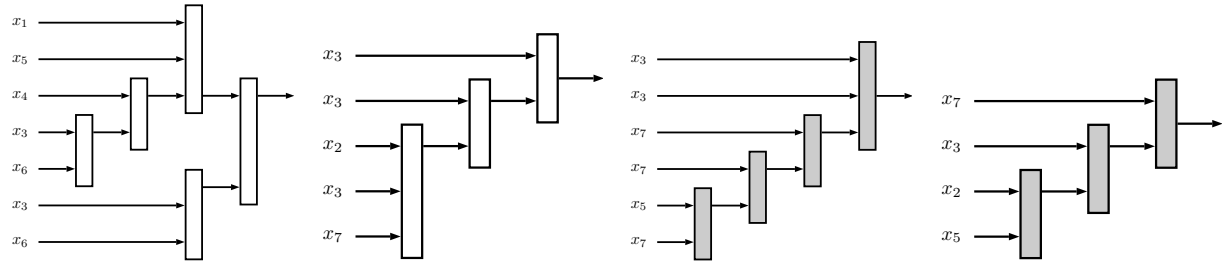
PERFORMANCE EVALUATION ON ABALONE AGE PREDICTION PROBLEM (EXAMPLE-4)

(a) Performance Statistics (10 Repetitions)

		T1HFIT^{S}	T1HFIT^{M}	T2HFIT^{S}	T2HFIT^{M}
E_n	Best	2.1097	2.2857	2.1154	2.1275
	Mean	2.3267	2.4284	2.2597	2.2404
	STD	0.1534	0.1079	0.1478	0.0627
r_n	Best	0.76	0.71	0.76	0.75
	Mean	0.688	0.655	0.710	0.716
	STD	0.0490	0.0347	0.0481	0.0204
E_t	Best	2.1260	2.3480	2.1824	2.1428
	Mean	2.3644	2.4843	2.3808	2.3533
	STD	0.1448	0.1029	0.1676	0.1127
r_t	Best	0.76	0.71	0.76	0.75
	Mean	0.688	0.655	0.710	0.716
	STD	0.0490	0.0347	0.0481	0.0204
$c(\mathbf{w})$	Best	20	20	144	72
	Mean	77.6	46.4	188.4	152.9
Time	Best	45.53	40.22	88.5	65.02
	Mean	83.33	70.16	213.5	192

(b) Performance Comparison

	Algorithm	E_n	E_t	$c(\mathbf{w})$
Type–1	HS	–	3.1600	–
	General	–	3.1500	–
	CCL	–	2.6500	–
	Chen&Cheng	–	2.5900	–
	T1HFIT^{S}	2.1097	2.1260	124
	T1HFIT^{M}	2.2857	2.3480	84
Type–2	RIT2NFS-WB	2.4047	2.1346	131
	McIT2FIS-UM	2.3481	1.8740	115
	SEIT2FNN	2.3388	2.4330	140
	McIT2FIS-US	2.3357	1.8387	115
	T2HFIT^{S}	2.1154	2.1824	226
	T2HFIT^{M}	2.1275	2.1428	108



(a) T1HFIT^{S}: $E_n = 2.1097$ (b) T1HFIT^{M}: $E_n = 2.2857$ (c) T2HFIT^{S}: $E_n = 2.1154$ (d) T2HFIT^{M}: $E_n = 2.1275$

Fig. 10. Example–4: designed HFIT, where the shaded nodes are T2FIS.

E. Example 5—Box-Jenkins Gas Furnace Problem

In this example, the Box and Jenkins gas furnace dataset that was taken from [96], which has 296 data samples. The objective of this example was to predict the CO_2 concentration from the gas-flow rate. The gas furnace system is modeled using a series, which is of the form: $y(k) = f(y(k-1), u(k-4)$. For the training of the proposed models, as mentioned in [24],

TABLE VIII

PERFORMANCE EVALUATION ON BOX-JENKINS GAS CONCENTRATION PROBLEM (EXAMPLE-5)

(a) Performance Statistics (10 Repetitions)

		T1HFIT$^{\mathrm{S}}$	T1HFIT$^{\mathrm{M}}$	T2HFIT$^{\mathrm{S}}$	T2HFIT$^{\mathrm{M}}$
E_n	Best	0.246	0.280	0.256	0.275
	Mean	0.303	0.344	0.291	0.301
	STD	0.036	0.043	0.023	0.033
r_n	Best	0.97	0.97	0.97	0.97
	Mean	0.959	0.947	0.963	0.960
	STD	0.010	0.013	0.006	0.010
$c(\mathbf{w})$	Best	40	40	72	72
	Mean	132.8	58.4	286	167.4
Time	Best	5.41	4.22	9.82	6.31
	Mean	11.84	4.76	20.67	11.15

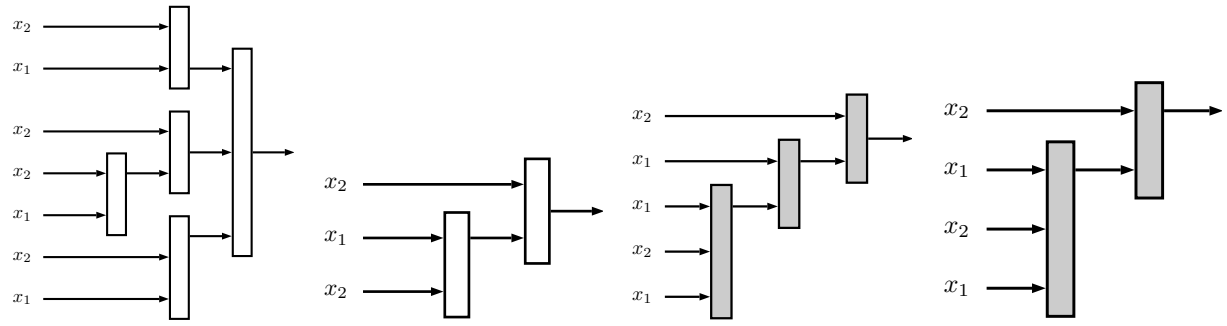
(b) Performance Comparison

	Algorithm	E_n	$c(\mathbf{w})$
Type−1	T1-NFS	0.4074	–
	GNN^{1}	0.3114	–
	GNN^{2}	0.2983	–
	T1HFIT$^{\mathrm{S}}$	0.2455	124
	T1HFIT$^{\mathrm{M}}$	0.2838	40
Type−2	SEIT2FNN	0.2690	152
	RIT2NFS-WB	0.3527	90
	McIT2FIS-UM	0.3139	48
	McIT2FIS-US	0.3181	48
	T2HFIT$^{\mathrm{S}}$	0.2767	154
	T2HFIT$^{\mathrm{M}}$	0.2840	72

100% (296 patterns) of the samples were used. To show an average performance ability of the proposed algorithms, the training process was also repeated ten times, and the collected results are summarized in Table VIII(a). The performances of the proposed algorithms (the best results) were compared with the best performances of the algorithms reported in the literature (Table VIII(b)).

To compare the performance of the algorithms, the results of T1-NFS and GNN were collected from [17], and the results of SEIT2FNN, RIT2NFS-WB, McIT2FIS-UM, and McIT2FIS-US were collected from [24]. As reported in Table VIII(b), the proposed algorithms clearly outperformed the algorithms T1-NFS, GNN^{1}, and GNN^{2} in the case of T1FIS comparisons and algorithms SEIT2FNN, RIT2NFS-WB, McIT2FIS-UM, and McIT2FIS-US in the case of T2FIS comparisons.

For T2FIS, the proposed algorithm T2HFIT$^{\mathrm{M}}$ provided a training RMSE $E_n = 0.284$, which was slightly lower than the training RMSE $E_n = 0.269$ of SEIT2FNN. However, the parameter count of T2HFIT$^{\mathrm{M}}$ was 72 compared to 152 of SEIT2FNN. Additionally, despite being a population based algorithm, T2HFIT$^{\mathrm{M}}$ takes 6.31 minutes for the training, whereas SEIT2FNN takes 604.66 minutes for the training [17]. Therefore, it may be concluded that, for example 5, T2HFIT$^{\mathrm{M}}$ performed the best. The best-performing models are illustrated in Fig. 11.



(a) T1HFIT$^{\text{S}}$: $E_n = 0.2455$ (b) T1HFIT$^{\text{M}}$: $E_n = 0.2838$ (c) T2HFIT$^{\text{S}}$: $E_n = 0.2767$ (d) T2HFIT$^{\text{M}}$: $E_n = 0.2840$

Fig. 11. Example–5: designed HFIT, where the shaded nodes are T2FIS.

F. Example 6—Poly (lactic-co-glycolic acid) (PLGA) micro- and nanoparticle dissolution rate prediction

This example illustrates a pharmaceutical industry problem related to PLGA dissolution profile prediction, which is a complex problem since a vast number of factors governs its dissolution rate profile and it has a high noise and redundancy because the dataset was obtained from various experimental measurements and instruments. As per the dataset provided in [97], [98], this problem has 747 samples and a total of 300 input features, which influence the PLGA protein particle's dissolution rate [99]. The input features are categorized into five groups: protein descriptor, formulation characteristics, plasticizer, emulsifier, and time delay, which has 85, 17, 98, 99, and 1 features, respectively.

The description of each feature group is as follows: 1) The protein descriptors (85 features) describe the type of molecules and proteins used in the drug's manufacturing. 2) The formulation characteristics (17 features) describe the molecular properties, such as molecular weight, particle size, etc., of the molecules and proteins. 3) The plasticizer (98 features) describes properties, such as fluidity of the material used. 4) The emulsifier (99 features) describes the stabilizing properties of the material used in the drug's manufacturing. 5) The time delay (1 feature) represents the time taken to dissolve/dissolute a sample drug.

The PLGA dissolution profile prediction is a significant problem since it plays a crucial role in the medical application and toxicity evaluation of PLGA-based microparticles dosages [100]. Moreover, PLGA microparticles are important diluents, which are used for producing drugs in their correct dosage form. It is also used as a filler, as an excipient, and as an active pharmaceutical ingredient because it acts as a catalyst for drug absorption/dissolution/solubility [101].

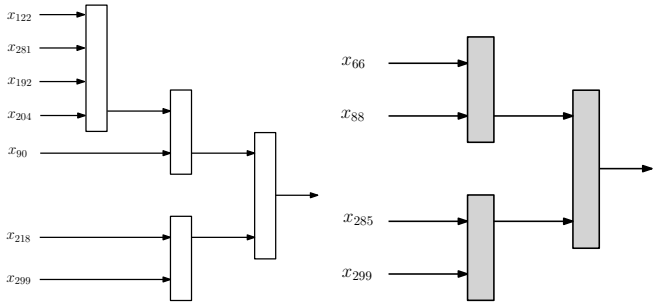
TABLE IX
EXAMPLE 6: PERFORMANCE COMPARISON

Algorithm	Ref.	RMSE E_t	No. of features
MLP	[97]	14.3	17
HFIT	[102]	13.2	15
REP Tree	[98]	13.3	15
GPR	[98]	14.9	15
MLP	[98]	15.2	15
MLP	[97]	15.4	11
T1HFIT$^{\mathrm{M}}$	present work	18.6	7
T2HFIT$^{\mathrm{M}}$	present work	15.2	4

Therefore, PLGA dissolution is a widely studied research problem in pharmaceutical manufacturing and powder technology.

Using the parameter setting mentioned in Table I and using 10-fold cross-validation, the proposed algorithm T1HFIT$^{\mathrm{M}}$ was able to select seven input features and was able to approximate a test RMSE of $E_t = 18.66$. The selected features were: phase polyvinyl alcohol Mw (x_{90}), ASA (x_{122}), pH 8 msdon (x_{192}), aromatic bond count (x_{204}), a(xx) (x_{218}), pH 12 msacc (x_{281}), time days (x_{299}). Similarly, the proposed algorithm T2HFIT$^{\mathrm{M}}$ was able to approximate a test RMSE of $E_t = 15.259$ with only four input features: aromatic atom count (x_{66}), phase polyvinyl alcohol concentration inner phase (x_{88}), pH 1 msdon (x_{285}), time days (x_{299}). Additionally, T2HFIT$^{\mathrm{M}}$ provided a simple model (i.e., $c(\mathbf{w}) = 108$) compared to T2HFIT$^{\mathrm{M}}$ that had model complexity $c(\mathbf{w}) = 156$. Moreover, T2HFIT$^{\mathrm{M}}$ takes 7.16 minutes of training time compared to the 45.7 minutes of T1HFIT$^{\mathrm{M}}$. This difference in time is due to the difference between the number of input features being selected by T2HFIT$^{\mathrm{M}}$ and T1HFIT$^{\mathrm{M}}$.

Feature reduction is a significant task since it reduces drug's manufacturing cost. Table IX shows a comparison of the proposed T1HFIT$^{\mathrm{M}}$ and T2HFIT$^{\mathrm{M}}$ with algorithms such as multilayer perceptron (MLP), reduced error pruning tree (REP Tree), heterogeneous flexible neural tree (HFIT), and Gaussian process regression (GPR). It is evident from the results that the proposed algorithm approximates the PLGA dissolution profile with a lower number of features, and its approximation error was competitive with the performance of other algorithms. Fig. 12 illustrates the obtained models for the prediction of the PLGA dissolution profile.



(a) T1HFIT$^{\mathrm{M}}$: $E_t = 18.66$ (b) T2HFIT$^{\mathrm{M}}$: $E_t = 15.25$

Fig. 12. Example–6: Designed HFIT, where the shaded nodes are T2FIS.

VI. Discussion

The proposed HFIT algorithms T1HFIT$^{\mathrm{S}}$, T1HFIT$^{\mathrm{M}}$, T2HFIT$^{\mathrm{S}}$, and T2HFIT$^{\mathrm{M}}$ were evaluated through six examples, including a real-world example from the pharmaceutical industry. Performance of the proposed algorithms was compared with algorithms that offer fuzzy system's structural optimization (e.g., SEIT2FNN, SONFIN, SaFIN, TSCIT2FNN, etc.), hierarchical fuzzy system design (e.g., IFRS, H-TS-FS, etc.), dynamic fuzzy system design (e.g., DENFIS, D-FNN, etc.), and so forth. The obtained results illustrate the efficiency of the proposed algorithms in comparison to the algorithms collected from the literature. Such performance was obtained by using the parameter setting mentioned in Table I. Moreover, a comparison using noisy data [example 2, case 2 (Section V-B1)] has proved the approximation efficiency of the proposed algorithms over other algorithms. The HFIT algorithms not only offer solutions with high accuracy (low approximation error), but they also provide the solutions with low complexity.

The number of clusters needs to be predetermined in the algorithms that use a cluster-based partitioning of the input space and to define fuzzy sets. On the contrary, the proposed HFIT uses a only two partitions for each inputs and automatically defines fuzzy sets by using the dynamics of the evolutionary process. Such ability is particularly significant for the predictive modeling of problems like example 6 (Section V-F) that has a large number of input features. It would be a difficult task for fuzzy-NN-based algorithms (e.g., SONFIN, SEIT2FNN, McIT2FIS, etc.) to design a network-like structure to solve a high-dimensional problem (e.g., example–6 that has 300 input features), whereas the proposed HFIT solves example–6 with satisfactory accuracy and low model's complexity. Section IV-D show that HFIT has several qualities that set it apart from many algorithms mentioned in this work.

In Section V, a comprehensive study of the comparative results of the proposed algorithms was presented. It was observed that the proposed HFIT-based algorithms gave better performance than the other algorithms collected from the literature. For example, in the case of example–1 T1HFIT^{M} provided better RMSE with a lower parameter count. Additionally, T2HFIT^{M} offered an RMSE (i.e., 0.0028) with a low complexity (i.e., 72) in comparison to SEIT2FNN that gave an RMSE of 0.0022 with a model complexity of 84.

Similarly, for example-2, T2HFIT^{M} offered a competitive RMSE (i.e., 0.0058) in comparison to SEIT2FNN[2] that gave an RMSE of 0.0053. Additionally, in the noisy dataset comparison, the proposed T2HFIT^{M} provided better training RMSEs with lower model's complexities when compared to many of the recently proposed T2FIS algorithms, such as SEIT2FNN, IT2FNN-SVR, and T2FLS-G. Moreover, the models developed by the proposed algorithm adapted its structure in each instance of noisy dataset experiments; whereas, the other models had a fixed structure in each instance of their experiments (Table V). Therefore, the proposed algorithm was able to accommodate the variance in noise more precisely than the other models.

With example–3, example–4, and example–5, the proposed type-1 HFIT surpassed all the other algorithms. Whereas, type-2 HFIT performed competitively with algorithms such as RIT2NFS-WB, McIT2FIS, and SEIT2FNN. It was observed that the training RMSE of T2HFIT^{M} for example–3 was as per with RIT2NFS-WB, but the complexity of the proposed T2HFIT^{M} was much less than RIT2NFS-WB. For example–4, T2HFIT^{M} outperformed all its counterparts in both accuracy and complexity. For example–5, the training RMSE of T2HFIT^{M} was close to SEIT2FNN, but on model complexity and training time, T2HFIT^{M} outperformed SEIT2FNN by a comfortable margin. Therefore, it may be concluded that the proposed HFIT version performed efficiently against other algorithms found in the literature.

The proposed HFIT is a population-based algorithm. Therefore, it should naturally take more training time than a single solution based algorithm. In addition to that, the training time depends on several factors such as the programming language used, the type of platform, the hardware configuration of the machine, the method of data feeding during the training, etc. Therefore, training time comparison is limited. However, by comparing the training time of the proposed algorithms with the training times of some of other algorithms (training time of only a few algorithms is reported in the literature), the following was observed: 1) in the case of example–1, the proposed T2HFIT^{M} was found competitive with other algorithms, 2) in the case of example–5, T2HFIT^{M} outperformed SEIT2FNN, 3) in the case of example–6, which has 747 samples and

TABLE X

PERFORMANCE SUMMARY ON BENCHMARK EXAMPLES: SINGLE OBJECTIVE VERSUS MULTIOBJECTIVE AND TYPE-1 VERSUS TYPE-2

	Single Objective				Multibjective			
	T1HFIT$^{\mathrm{S}}$		T2HFIT$^{\mathrm{S}}$		T1HFIT$^{\mathrm{M}}$		T2HFIT$^{\mathrm{M}}$	
Example	E_n	$c(\mathbf{w})$	E_n	$c(\mathbf{w})$	E_n	$c(\mathbf{w})$	E_n	$c(\mathbf{w})$
1	0.018	57.2	0.012	152.0	0.025	34.4	0.018	90.4
2	0.034	71.7	0.041	203.4	0.033	57.6	0.022	129.5
3	2.711	132.0	2.469	224.0	2.603	78.8	2.405	204.4
4	2.326	77.6	2.259	188.4	2.428	46.4	2.242	152.9
5	0.303	138.8	0.291	286.0	0.344	58.4	0.301	167.4
6	24.32	220.0	16.499	208.0	17.448	156.0	14.352	108.0
Average	4.952	116.2	3.595	210.3	3.814	71.9	3.223	142.1

300 input features, the proposed T2HFIT$^{\mathrm{M}}$ takes only about 7.16 minutes, which is remarkable.

For example–6, the proposed T2HFIT$^{\mathrm{M}}$ was more efficient than T1HFIT$^{\mathrm{M}}$ because T2HFIT$^{\mathrm{M}}$ was capable of accommodating noisy information more efficiently than T1HFIT$^{\mathrm{M}}$. This is evident from the fact that the average RMSE of T2HFIT$^{\mathrm{M}}$ was 16.64, and the average RMSE of T1HFIT$^{\mathrm{M}}$ was 22.36. Hence, the proposed T2HFIT model, which is relied on interval type-2 MFs, is worth considering in such high-dimensional and noisy application problems.

A comparison between single objective and multiobjective summarized in Table X suggests that the multiobjective approach has performance superiority over the single objective because multiobjective gives a competitively better approximation error with lower model complexity in both type-1 and type2 cases compared to single objective. Additionally, it can be observed that type-2 HFIT offers better approximation error against type-1 HFIT.

Since HFIT algorithms were developed using the evolutionary process, the quality of their performance is subjected to carefully setting of the parameters mentioned in Table I. Hence, the results of the algorithms mentioned in this work may be further improved upon by choosing different sets of parameters; however, this is a trial-and-error process. For example, the feature selection, i.e., the number of inputs into a node (a fuzzy subsystem) is proportional to the setting of the maximum inputs into an node. Similarly, the hierarchy (number of layers) in an HFIT is proportional to the setting of the maximum depth of a tree. Therefore, HFIT's complexity can be controlled using these parameters. Additionally, the parameters of MOGP and DE, such as their

population size, crossover probability, mutation probability, etc., influence HFIT's performance.

VII. Conclusions

Using a fuzzy inference system (FIS) for data mining inherently requires a multiobjective solution and the proposed multiobjective design for a hierarchical fuzzy inference tree (HFIT) stands as a viable option that constructs a tree-like model whose nodes are low-dimensional FIS. The proposed HFIT was developed for both type-1 and type-2 FIS and each node in HFIT implements a Takagi–Sugeno–Kang model. Both type-1 and type-2 FIS were studied in the scope of single objective and multiobjective optimization using genetic programming. Hence, four versions of HFIT were studied: T1HFIT$^{\mathrm{S}}$, T1HFIT$^{\mathrm{M}}$, T2HFIT$^{\mathrm{S}}$, and T2HFIT$^{\mathrm{M}}$. The parameters of the membership functions and the consequent parts of the rules were optimized using a differential evolution algorithm. HFIT's optimization procedure was a two-phase evolutionary optimization approach, in which structure optimization and parameter optimization were applied one-by-one until a formidable solution was obtained. The approximation ability of the proposed HFIT was theoretically examined. As a result of that four distinguished quality of HFIT was discovered: adaptation in structure, diverse rule generation, automatic fuzzy set selection, minimal feature drive structure formation. A comprehensive performance comparison was performed for evaluating the efficiency of the proposed HFIT. The performance of the proposed HFIT algorithm was found to be efficient and competitive compared to the algorithms collected from the literature. HFIT provided competitive approximation compared to other algorithms and simultaneously it produced less complex models. Additionally, HFIT performs feature selection and automatic structure design, which is a necessary for solving high-dimensional problems.

References

[1] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst. Man Cybern.*, vol. 15, no. 1, pp. 116–132, 1985.

[2] L. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, no. 3, pp. 338 – 353, 1965.

[3] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning," *Inf. Sci.*, vol. 8, no. 3, pp. 199–249, 1975.

[4] N. N. Karnik, J. M. Mendel, and Q. Liang, "Type-2 fuzzy logic systems," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 6, pp. 643–658, 1999.

[5] H. A. Hagras, "A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 4, pp. 524–539, 2004.

[6] L. B. Booker, "Intelligent behavior as an adaptation to the task environment," Ph.D. dissertation, University of Michigan, Ann Arbor, MI, USA, 1982.

[7] H. Ishibuchi, T. Nakashima, and T. Kuroda, "A hybrid fuzzy genetics-based machine learning algorithm: hybridization of Michigan approach and Pittsburgh approach," in *1999 Int. Conf. Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conf. Proc.*, vol. 1, pp. 296–301.

[8] S. F. Smith, "A learning system based on genetic adaptive algorithms," Ph.D. dissertation, University of Pittsburgh, Pittsburgh, PA, USA, 1980.

[9] J.-S. R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Trans. Syst. Man Cybern.*, vol. 23, no. 3, pp. 665–685, 1993.

[10] S. Wu and M. J. Er, "Dynamic fuzzy neural networks–a novel approach to function approximation," *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 30, no. 2, pp. 358–364, 2000.

[11] K. D. Sharma, A. Chatterjee, and A. Rakshit, "A hybrid approach for design of stable adaptive fuzzy controllers employing lyapunov theory and particle swarm optimization," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 2, pp. 329–342, 2009.

[12] S.-C. Huang, M.-K. Jiau, and C.-H. Lin, "Optimization of the carpool service problem via a fuzzy-controlled genetic algorithm," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 5, pp. 1698–1712, 2015.

[13] O. Castillo and P. Melin, "Optimization of type-2 fuzzy systems based on bio-inspired methods: a concise review," *Inf. Sci.*, vol. 205, pp. 1–19, 2012.

[14] A. Fernández, V. López, M. J. del Jesus, and F. Herrera, "Revisiting evolutionary fuzzy systems: Taxonomy, applications, new trends and challenges," *Knowledge-Based Syst.*, vol. 80, pp. 109–121, 2015.

[15] C.-F. Juang and C.-T. Lin, "An online self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 1, pp. 12–32, 1998.

[16] C.-F. Juang and Y.-W. Tsao, "A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 6, pp. 1411–1424, 2008.

[17] C.-F. Juang and K.-J. Juang, "Reduced interval type-2 neural fuzzy system using weighted bound-set boundary operation for computation speedup and chip implementation," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 3, pp. 477–491, 2013.

[18] N. K. Kasabov and Q. Song, "DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 144–154, 2002.

[19] S. W. Tung, C. Quek, and C. Guan, "SaFIN: A self-adaptive fuzzy inference network," *IEEE Trans. Neural Netw*, vol. 22, no. 12, pp. 1928–1940, 2011.

[20] Y.-Y. Lin, J.-Y. Chang, N. R. Pal, and C.-T. Lin, "A mutually recurrent interval type-2 neural fuzzy system (MRIT2NFS) with self-evolving structure and parameters," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 3, pp. 492–509, 2013.

[21] Y.-Y. Lin, J.-Y. Chang, and C.-T. Lin, "A TSK-type-based self-evolving compensatory interval type-2 fuzzy neural network (TSCIT2FNN) and its applications," *IEEE Trans. Ind. Electron.*, vol. 61, no. 1, pp. 447–459, 2014.

[22] Y.-Y. Lin, S.-H. Liao, J.-Y. Chang, and C.-T. Lin, "Simplified interval type-2 fuzzy neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 5, pp. 959–969, 2014.

[23] A. Bouchachia and C. Vanaret, "GT2FC: an online growing interval type-2 self-learning fuzzy classifier," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 4, pp. 999–1018, 2014.

[24] A. K. Das, K. Subramanian, and S. Sundaram, "An evolving interval type-2 neurofuzzy inference system and its metacognitive sequential learning algorithm," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 6, pp. 2080–2093, 2015.

[25] G. Raju, J. Zhou, and R. A. Kisner, "Hierarchical fuzzy control," *Int. J. Control*, vol. 54, no. 5, pp. 1201–1216, 1991.

[26] M. Brown, K. Bossley, D. Mills, and C. Harris, "High dimensional neurofuzzy systems: overcoming the curse of dimensionality," in *Proc. 1995 IEEE Int. Fuzzy Syst., 1995. Int. Jt. Conf. of the 4th Int. Conf. Fuzzy Syst. and The 2nd Int. Fuzzy Eng. Symp.*, vol. 4, pp. 2139–2146.

[27] L.-X. Wang, "Analysis and design of hierarchical fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 5, pp. 617–624, 1999.

[28] M. R. Delgado, F. Von Zuben, and F. Gomide, "Hierarchical genetic fuzzy systems," *Inf. Sci.*, vol. 136, no. 1, pp. 29–52, 2001.

[29] M.-L. Lee, H.-Y. Chung, and F.-M. Yu, "Modeling of hierarchical fuzzy systems," *Fuzzy Sets Syst.*, vol. 138, no. 2, pp. 343–361, 2003.

[30] X.-J. Zeng and J. A. Keane, "Approximation capabilities of hierarchical fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 13, no. 5, pp. 659–672, 2005.

[31] V. Torra, "A review of the construction of hierarchical fuzzy systems," *Int. J. Intell. Syst.*, vol. 17, no. 5, pp. 531–543, 2002.

[32] M. G. Joo and J. S. Lee, "A class of hierarchical fuzzy systems with constraints on the fuzzy rules," *IEEE Trans. Fuzzy Syst.*, vol. 13, no. 2, pp. 194–203, 2005.

[33] A. Fernández, M. J. del Jesus, and F. Herrera, "Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets," *Int. J. Approximate Reasoning*, vol. 50, no. 3, pp. 561–577, 2009.

[34] C.-L. Hwang, C.-C. Chiang, and Y.-W. Yeh, "Adaptive fuzzy hierarchical sliding-mode control for the trajectory tracking of uncertain underactuated nonlinear dynamic systems," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 2, pp. 286–299, 2014.

[35] Y. Chen, B. Yang, A. Abraham, and L. Peng, "Automatic design of hierarchical takagi–sugeno type fuzzy systems using evolutionary algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 3, pp. 385–397, 2007.

[36] R. Salustowicz and J. Schmidhuber, "Probabilistic incremental program evolution," *Evol. Comput.*, vol. 5, no. 2, pp. 123–141, 1997.

[37] A. Mohammadzadeh, O. Kaynak, and M. Teshnehlab, "Two-mode indirect adaptive control approach for the synchronization of uncertain chaotic systems by the use of a hierarchical interval type-2 fuzzy neural network," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 5, pp. 1301–1312, 2014.

[38] H. Ishibuchi, "Multiobjective genetic fuzzy systems: review and future research directions," in *IEEE Int. Fuzzy Syst. Conf., 2007. FUZZ-IEEE 2007*, pp. 1–6.

[39] H. Ishibuchi, T. Murata, and I. Türkşen, "Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems," *Fuzzy Sets Syst.*, vol. 89, no. 2, pp. 135–150, 1997.

[40] R. Alcalá, M. J. Gacto, F. Herrera, and J. Alcalá-Fdez, "A multi-objective genetic algorithm for tuning and rule selection to obtain accurate and compact linguistic fuzzy rule-based systems," *Int. J. Uncertainty Fuzziness Knowledge Based Syst.*, vol. 15, no. 05, pp. 539–557, 2007.

[41] O. Cordón, "A historical review of evolutionary learning methods for Mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems," *Int. J. Approximate Reasoning*, vol. 52, no. 6, pp. 894–913, 2011.

[42] S. Guillaume, "Designing fuzzy inference systems from data: an interpretability-oriented review," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 3, pp. 426–443, 2001.

[43] H. Ishibuchi and Y. Nojima, "Analysis of interpretability-accuracy trade-off of fuzzy systems by multiobjective fuzzy genetics-based machine learning," *Int. J. Approximate Reasoning*, vol. 44, no. 1, pp. 4–31, 2007.

[44] M. J. Gacto, R. Alcalá, and F. Herrera, "Adaptation and application of multi-objective evolutionary algorithms for rule reduction and parameter tuning of fuzzy rule-based systems," *Soft Comput.*, vol. 13, no. 5, pp. 419–436, 2009.

[45] C. J. Carmona, P. González, M. J. del Jesus, and F. Herrera, "NMEEF-SD: non-dominated multiobjective evolutionary algorithm for extracting fuzzy rules in subgroup discovery," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 5, pp. 958–970, 2010.

[46] A. B. Cara, C. Wagner, H. Hagras, H. Pomares, and I. Rojas, "Multiobjective optimization and comparison of nonsingleton type-1 and singleton interval type-2 fuzzy logic systems," *IEEE Trans. Fuzzy syst.*, vol. 21, no. 3, pp. 459–476, 2013.

[47] H. Wang, S. Kwong, Y. Jin, W. Wei, and K.-F. Man, "Multi-objective hierarchical genetic algorithm for interpretable fuzzy rule-based knowledge extraction," *Fuzzy Sets Syst.*, vol. 149, no. 1, pp. 149–186, 2005.

[48] R. Munoz-Salinas, E. Aguirre, O. Cordón, and M. García-Silvente, "Automatic tuning of a fuzzy visual system using evolutionary algorithms: single-objective versus multiobjective approaches," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 2, pp. 485–501, 2008.

[49] R. Alcalá, P. Ducange, F. Herrera, B. Lazzerini, and F. Marcelloni, "A multiobjective evolutionary approach to concurrently learn rule and data bases of linguistic fuzzy-rule-based systems," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 5, pp. 1106–1122, 2009.

[50] M. Antonelli, P. Ducange, B. Lazzerini, and F. Marcelloni, "Learning knowledge bases of multi-objective evolutionary fuzzy systems by simultaneously optimizing accuracy, complexity and partition integrity," *Soft Comput.*, vol. 15, no. 12, pp. 2335–2354, 2011.

[51] M. Antonelli, P. Ducange, and F. Marcelloni, "Genetic training instance selection in multiobjective evolutionary fuzzy systems: A coevolutionary approach," *IEEE Trans. Fuzzy Syst.*, vol. 20, no. 2, pp. 276–290, 2012.

[52] M. Fazzolari, R. Alcala, Y. Nojima, H. Ishibuchi, and F. Herrera, "A review of the application of multiobjective evolutionary fuzzy systems: Current status and further directions," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 1, pp. 45–65, 2013.

[53] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza, *A field guide to genetic programming*. Lulu. com, 2008.

[54] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, 2009.

[55] J. M. Mendel, "On KM algorithms for solving type-2 fuzzy set problems," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 3, pp. 426–446, 2013.

[56] Y. Chen, B. Yang, J. Dong, and A. Abraham, "Time-series forecasting using flexible neural tree model," *Inf. Sci.*, vol. 174, no. 3, pp. 219–235, 2005.

[57] Y. Jin, B. Sendhoff, and E. Körner, "Evolutionary multi-objective optimization for simultaneous generation of signal-type and symbol-type representations," in *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science, vol. 3410. Springer, 2005, pp. 752–766.

[58] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Parallel Problem Solving from Nature PPSN VI*, ser. Lecture Notes in Computer Science. Springer, 2000, vol. 1917, pp. 849–858.

[59] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 2nd ed. Heidelberg: Springer, 2003.

[60] V. K. Ojha, A. Abraham, and V. Snášel, "Ensemble of heterogeneous flexible neural trees using multiobjective genetic programming," *Appl. Soft Comput.*, vol. 52, pp. 909–924, 2017.

[61] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Global Optim.*, vol. 39, no. 3, pp. 459–471, 2007.

[62] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, 2007.

[63] S. Das, S. S. Mullick, and P. Suganthan, "Recent advances in differential evolution–an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[64] J. Snyman, *Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms*. Springer, 2005, vol. 97.

[65] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[66] S. S. Haykin, *Kalman filtering and Neural Networks*. Wiley Online Library, 2001.

[67] D. E. Goldberg and P. Segrest, "Finite markov chain analysis of genetic algorithms," in *Proc. of the 2nd Int. Conf. on Genetic Algorithms and Their Appl.* Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1987, pp. 1–8.

[68] A. Szalas and Z. Michalewicz, "Contractive mapping genetic algorithms and their convergence," Dept. of Computer Science, University of North Carolina at Charlotte, Tech. Rep. Technical Report 006-1993, 1993.

[69] S. Banach, "Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales (in French)," *Fund. Math.*, vol. 3, no. 1, pp. 133–181, 1922.

[70] J. Dixmier, *General Topology*. Springer, 1984.

[71] L. Altenberg, "The evolution of evolvability in genetic programming," in *Advances in Genetic Programming*, K. E. Kinnear Jr., Ed., vol. 3. MIT Press, 1994, pp. 47–74.

[72] A. N. Kolmogorov, "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition," *Transl. Amer. Math. Soc.*, vol. 28, no. 2, pp. 55–59, 1963.

[73] F. Xue, A. C. Sanderson, and R. J. Graves, "Modeling and convergence analysis of a continuous multi-objective differential evolution algorithm," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 1. IEEE, 2005, pp. 228–235.

[74] J. Zhang and A. C. Sanderson, "Theoretical analysis of differential evolution," in *Adaptive Differential Evolution*. Springer, 2009, pp. 15–38.

[75] Z. Hu, S. Xiong, Q. Su, and X. Zhang, "Sufficient conditions for global convergence of differential evolution algorithm," *J. Appl. Math.*, vol. 2013, 2013.

[76] S. Paul and S. Kumar, "Subsethood-product fuzzy neural inference system (SuPFuNIS)," *IEEE Trans. Neural Netw*, vol. 13, no. 3, pp. 578–599, 2002.

[77] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, Dec 1997.

[78] N. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning," *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 31, no. 6, pp. 902–918, 2001.

[79] C.-J. Lin and C.-T. Lin, "An ART-based fuzzy adaptive learning control network," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 4, pp. 477–496, 1997.

[80] Y.-Q. Zhang, B. Jin, and Y. Tang, "Granular neural networks with evolutionary interval learning," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 2, pp. 309–319, 2008.

[81] J. Kim and N. Kasabov, "HyFIS: adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems," *Neural Netw.*, vol. 12, no. 9, pp. 1301–1319, 1999.

[82] J.-C. Duan and F.-L. Chung, "Multilevel fuzzy relational systems: structure and identification," *Soft Comput.*, vol. 6, no. 2, pp. 71–86, 2002.

[83] K. B. Cho and B. H. Wang, "Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction," *Fuzzy Sets Syst.*, vol. 83, no. 3, pp. 325–339, 1996.

[84] J.-H. Chiang and P.-Y. Hao, "Support vector learning mechanism for fuzzy rule-based modeling: a new approach," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 1, pp. 1–12, 2004.

[85] S. W. Tung, C. Quek, and C. Guan, "eT2FIS: an evolving type-2 neural fuzzy inference system," *Inf. Sci.*, vol. 220, pp. 124–148, 2013.

[86] C.-F. Juang, R.-B. Huang, and W.-Y. Cheng, "An interval type-2 fuzzy-neural network with support-vector regression for noisy regression problems," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 4, pp. 686–699, 2010.

[87] P. P. Angelov and D. P. Filev, "An approach to online identification of Takagi-Sugeno fuzzy models," *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 34, no. 1, pp. 484–498, 2004.

[88] J. M. Mendel, *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*. Upper Saddle River, NJ: Prentice-Hall, 2001.

[89] ——, "Computing derivatives in interval type-2 fuzzy logic systems," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 1, pp. 84–98, 2004.

[90] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw*, vol. 1, no. 1, pp. 4–27, 1990.

[91] M. Lichman, "UCI machine learning repository," 2013, accessed on: 01.05.2016. [Online]. Available: http://archive.ics.uci.edu/ml

[92] P. Baranyi, L. T. Kóczy, and T. T. D. Gedeon, "A generalized concept for fuzzy rule interpolation," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 6, pp. 820–837, 2004.

[93] Y.-C. Chang, S.-M. Chen, and C.-J. Liau, "Fuzzy interpolative reasoning for sparse fuzzy-rule-based systems based on the areas of fuzzy sets," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 5, pp. 1285–1301, 2008.

[94] Z. Huang and Q. Shen, "Fuzzy interpolation and extrapolation: A practical approach," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 1, pp. 13–28, 2008.

[95] S.-M. Chen and Y.-C. Chang, "Weighted fuzzy rule interpolation based on GA-based weight-learning techniques," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 4, pp. 729–744, 2011.

[96] G. E. P. Box and G. M. Jenkins, *Time Series Analysis, Forecasting and Control*. San Francisco, CA, USA: Holden–Day, 1976.

[97] J. Szlek, A. Pacławski, R. Lau, R. Jachowicz, and A. Mendyk, "Heuristic modeling of macromolecule release from PLGA microspheres," *Int. J. Nanomed.*, vol. 8, no. 1, pp. 4601–4611, 2013.

[98] V. K. Ojha, K. Jackowski, A. Abraham, and V. Snášel, "Dimensionality reduction, and function approximation of poly (lactic-co-glycolic acid) micro-and nanoparticle dissolution rate," *Int. J. Nanomed.*, vol. 10, pp. 1119 – 1129, 2015.

[99] C. E. Astete and C. M. Sabliov, "Synthesis and characterization of PLGA nanoparticles," *J. Biomater. Sci., Polym. Ed.*, vol. 17, no. 3, pp. 247–289, 2006.

[100] R. Langer and D. A. Tirrell, "Designing materials for biology and medicine," *Nature*, vol. 428, no. 6982, pp. 487–492, 2004.

[101] H. K. Makadia and S. J. Siegel, "Poly lactic-co-glycolic acid (PLGA) as biodegradable controlled drug delivery carrier," *Polymers (Basel)*, vol. 3, no. 3, pp. 1377–1397, 2011.

[102] V. K. Ojha, A. Abraham, and V. Snasel, "Ensemble of heterogeneous flexible neural tree for the approximation and feature-selection of poly (lactic-co-glycolic acid) micro-and nanoparticle," in *Proc. the 2nd Int. Afro-European Conf. for Ind. Adv. AECIA 2015*, 2016, pp. 155–165.